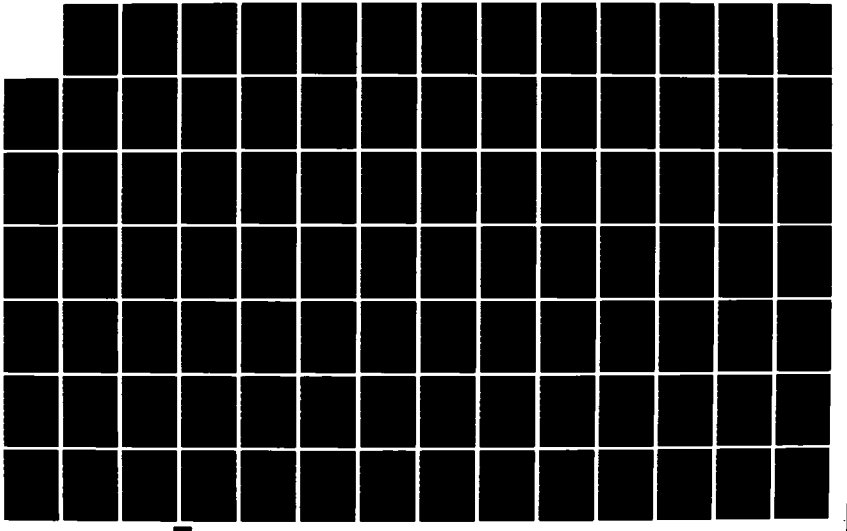
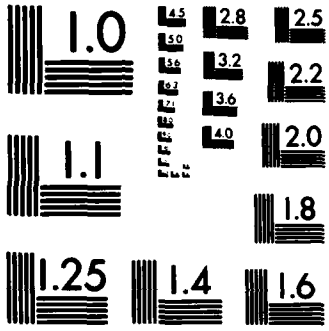
AD-A132 475 A COST CONSTRAINED SPEECH RECOGNITION SYSTEMS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH W F DAVIDSON AUG 83 AFIT/CI/NR-83-21T 1/2

UNCLASSIFIED F/G 17/2 NL





(1)

A COST CONSTRAINED SPEECH RECOGNITION SYSTEM

by

Walter F. Davidson

A Project Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF ENGINEERING

Approved:

Lester A. Gerhardt
Project Advisor

Rensselaer Polytechnic Institute
Troy, New York

August 1983

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFIT/CI/NR 83-21T	2. GOVT ACCESSION NO: AD-A132475
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): A Cost Constrained Speech Recognition Systems	5. TYPE OF REPORT & PERIOD COVERED: THESIS/DISSERTATION
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Walter F. Davidson	8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: AFIT STUDENT AT: Rensselaer Polytechnic Institute	10. PROGRAM ELEMENT PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: AFIT/NR, WPAFB OH 45433	12. REPORT DATE: Aug 1983
	13. NUMBER OF PAGES: 85
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):	15. SECURITY CLASS. (of this report): UNCLASS
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17

1 SEP 1983

Approved for public release: IAW AFR 190-17.
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED

DTIC COPY INSPECTED 2

Accession For	
NTIS GRA&I	☑
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT

This paper describes the design, construction, and evaluation of a microprocessor-based, cost-constrained word recognition system. The system utilizes seven bandpass filters, logarithmically spaced, followed by envelope detectors. The final algorithm uses eight uniformly spaced time slices, and used dynamic programming for time warping, with a weighted Tchebycheff distance. This system resulted in 98% correct recognition for the ten digits, $\emptyset$ - 9, of the training group, and 96% correct recognition for the control group.

The project demonstrated the necessity for an improvement gained with time warping. Rabiner's Unconstrained Endpoint Local Minima algorithm was used to perform the time warping. For the system used, it was found that a weighted Tchebycheff distance measure performed better than the Euclidean distance measure. The parameters were weighted inversely proportional to their variances. The results, however, were found to be relatively insensitive to the weighting coefficients.

The additional hardware required for a typical microprocessor system, costs under $150. The ability to build the hardware for such a low cost was due to the use of Reticon's Universal Active Filter R5620, which costs under $7.00 each.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

1.0 INTRODUCTION

The purpose of this project was to design and implement a speech recognition system for a limited vocabulary of isolated words. The goal was to produce a system that could do discrete word recognition on a microprocessor based system. With the present day proliferation of microprocessor systems, there are a wide variety of applications in which word recognition could be useful if the cost was low enough. These applications include numerical data entry and as a non-tactile input method for the physically handicapped.

There have been many attempts to build systems to do discrete word recognition. The most common present day systems use linear predictive coefficients and do all of their processing on a sampled version of the original voice waveform. This leads to very little additional external equipment to be added to the computer system, however, doing all the processing after sampling requires a large amount of computing power and is not practical for most microprocessor systems.

Since many other researchers have worked on the problem of limited vocabulary discrete word recognition, and obtained very good results, why try another approach? The goal of this project was to obtain a low cost recognition system that could be added to a typical microprocessor system. In addition to being low cost, it was desired to have a system that was easy to implement and did not require

any sophisticated test equipment to adjust. Due to these constraints it was necessary to limit the amount of post processing that was required. For this reason the approach chosen was to use external hardware to do prefiltering before the signal was sampled.

2.0 HISTORICAL REVIEW

The following table gives some statistics of past systems designed to do word recognition of the ten digits, 0 to 9. [1]

REFERENCE	SPEAKERS	NUMBER OF UTTERANCES	CORRECT
Martin, Grunza 1975	10	2400	99.7
Scott 1975	30	9300	98.0
Coler, et al 1977	20	20000	87.6
Nippon Electric 1978	4	2400	99.8

TABLE 1. PAST PERFORMANCES

"In general, scores of from 99% to as high as 99.9% correct recognition are possible in ideal laboratory conditions of no noise, adequate talker training, and consistent talking habits. However, actual field tests with ultimate users rarely come close to such high figures, and 97% is a high

(and barely adequate) accuracy level for most field conditions." [1] This project was not trying to improve the recognition rates that other researchers have obtained. The goal was to try to obtain similar results using a microprocessor based system with some low cost external hardware.

3.0 TECHNICAL DISCUSSION

Many distinguished researchers have proposed some, now classic, techniques for specific facets of pattern recognition. This project combined several of these classic techniques to obtain a word recognition procedure. Table 2 shows the major techniques tried for different levels of processing.

SAMPLING	WARPING	DISTANCE	CLASSIFICATION
UNIFORM SAMPLING	DYNAMIC PROGRAMMING WITH	TCHEBYCHEFF DISTANCE	K NEAREST NEIGHBOR
NON-UNIFORM SAMPLING	THRESHOLDING	EUCLIDEAN DISTANCE	FISHER DISCRIMINANT
			STANDARD DEVIATION WEIGHTING

TABLE 2. TECHNIQUES USED

Different combinations of these techniques, one from each column, were used in an attempt to obtain an optimum word recognition system. The following sections contain a

discussion of what word recognition is, an overview of the structure of the English language, and a brief summary of some of the major pattern recognition techniques employed in this project.

3.1.1 TYPES OF RECOGNITION

Systems that are using isolated words, words separated from other words by a period of silence, can be asked to perform one of three different types of speech recognition, WORD recognition, SPEAKER recognition, and WORD-SPEAKER recognition. If the system is responding to more than one speaker, the templates to be matched can be arranged in a matrix of i different words said by the j different speakers.

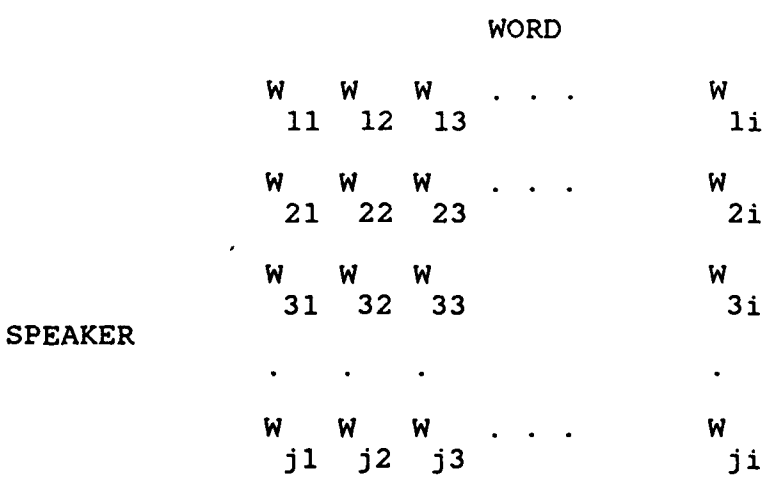


FIGURE 1. MATRIX REPRESENTATION

If the match results in a selection of the column number 1

through i, without regard to row, the routine is doing WORD recognition. In word recognition, the speaker who said the word is of no importance. For this reason, the matrix is treated as if the different rows are simply representing multiple utterances of the words spoken by the same speaker. If the answer from the routine is the row number 1 through j, the application is SPEAKER identification. In speaker identification or recognition, the word spoken is not important. In this case the matrix is treated as if the columns are simply multiple samples of the speakers voice. If the answer required is not only the word spoken, but also the speaker who said it, the answer must be both the row and column, and the application is WORD-SPEAKER recognition. As is quite apparent from the matrix representation, word-speaker recognition is the most difficult to accomplish, since it entails both of the other types of recognition.

3.1.2 PHONEMES

English can be described as a set of approximately 42 sounds called phonemes. These sounds can be further broken down into vowels, diphthongs, semivowels, and consonants. Each of the phonemes can be classified as either continuant or noncontinuant. Continuant sounds are those sounds that are produced by a fixed configuration of the vocal tract. [2]

CONTINUANT		NONCONTINUANT	
Vowels	Diphthongs		
IY	bEEt	AI	bUY
I	bIt	OI	bOY
E	bEt	AU	hOW
AE	bAt	EI	bAY
A	hOt	OU	bOAt
ER	bIRd	JU	yOU
UH	bUt		
OW	bOUght	Semivowels	
OO	bOOt	W	Wit
U	fOOt	L	Let
O	bOAt	R	Rent
		Y	You
Nasals			
M	Met	Stops	
N	Net	B	Bet
NG	siNG	D	Debt
		G	Get
Fricatives		P	Pet
voiced		T	Ten
V	Vat	K	Kit
TH	THing		
Z	Zoo	Whisper	
ZH	aZure	H	Hat
unvoiced		Affricates	
F	Fat	DZH	Judge
THE	THe	TSH	CHurch
S	Sat		
SH	SHut		

TABLE 3. PHONEMES

The approach for this project was to concentrate on the continuant phonemes. Continuant sounds are based on a fixed configuration of the vocal tract. Since the configuration of the vocal tract acts as a filter, a constant configuration will result in constant ratios of spectral components. Recognition is performed by obtaining the spectral energies

during these continuant phonemes, and matching these to the reference words with the same pattern of continuant phonemes. Even though noncontinuant phonemes are based on transitioning of the vocal tract, part of the phoneme will be based on a fixed vocal tract configuration. The proposed approach will therefore attempt to match all stationary sounds. The result is an attempt to match continuant phonemes and the stationary portions of noncontinuant phonemes.

3.1.3 DISTANCE MEASURES

The distance measure is a key element in the pattern matching algorithm. This system uses eight different features for pattern matching. A very important question is the weighted importance of each of the parameters. The averages and variances of these parameters must be estimated in order to calculate the required weighting of each of the parameters. The distance measures selected are that of a weighted Euclidean distance and a weighted Tchebycheff distance. The Euclidean distance measure is the proper measure to be used when the noise associated with the sample data is white and has a Gaussian distribution. The Euclidean distance measure is the proper distance measure to use with additive white noise, because the Euclidean distance measure, which is a square law detector, finds the intersection of the probability density functions when the

functions have equal variances, equal a priori probabilities, and Gaussian distributions. The purpose of the weighting is to normalize the different variances of the parameters. This procedure of weighting the parameters by the reciprocal of their variances is discussed by Duda and Hart. [3] This technique of weighting measurements inversely proportional to the variance estimates is a well known technique in Kalman Filtering to obtain a better estimate of a parameter in the presence of noise. To find the weighted Euclidean distance between two vectors X and Y, the square root is taken of the sum of the differences of each of the components, multiplied by the weighting factor for that component.

```
Euclidean = SQRT( (w1*(x1-y1)**2) + (w2*(x2-y2)**2) + ... )
Distance
```

To find the weighted Tchebycheff distance between two vectors X and Y, the sum is formed of the absolute value of the difference of the individual components multiplied by the weighting factor for that component.

```
Tchebycheff = w1*(|x1-y1|) + w2*(|x2-y2|) + ...
 Distance
```

These two distance measures are quite similar but the difference is that the Euclidean distance is a square law measure while the Tchebycheff distance is a first order measure. The Euclidean distance measure will perform better under conditions of white Gaussian noise. However, the Tchebycheff distance measure is often chosen, because most

microprocessors do not have a built in instruction to perform multiplication.

3.1.4 DYNAMIC PROGRAMING

Due to the inherent time variability of spoken words, it is necessary to use some form of warping in order to obtain a good match between two different utterances of the same word. Warping is the non-linear stretching or compressing of the word in order to obtain an optimal match with the reference template. Dynamic programming is normally used to perform this warping. There are many slightly different forms of dynamic programming, depending on the constraints placed on the problem.

The most naive approach is to treat each sequence as a uniform spring. In this method the end points are exactly matched by uniformly compressing or expanding the sequence. There are two main problems in trying to use this approach with this speech recognition algorithm. First, if the approach matches endpoints exactly, one must assume that the endpoints are the true endpoints. Isolated words normally have fairly well defined endpoints, but if the word starts or ends with a weak phoneme such as a fricative, the exact endpoint will not be well defined. Second, this method of uniform stretching, by definition, assumes that the increase or decrease in the number of points occurs uniformly across the sequence. In the Non-uniform algorithm, the points are

obtained non-uniformly from the detection of constant portions of the filtered envelopes, and this assumption is not valid. A small amount of noise during a portion of the word can cause points to be added or deleted from one portion of the sequence, while the remaining portions of the sequence are unaffected. This assumption of uniform stretching is not valid for the Uniform Algorithm either. In the Uniform algorithm, points are obtained at uniform time increments. A person is much more likely to draw out continuant phonemes, so again it is possible to add additional samples in part of the word without effecting the rest of the samples.

The dynamic programming method used to perform the warping for this project is a modification of Rabiner's Unconstrained Endpoint Local Minima (UELM) routine. [4] This method is a suboptimal form of dynamic programming. Instead of attempting to minimize the entire path, the UELM method only does local optimization. The advantage of this type of optimization is that there are a smaller number of possible paths to be examined. The reduction in the number of paths reduces the amount of required calculations, and consequentially the time that is needed for the routine to be performed.

In the UELM routine a match is found by finding the best fit of the next point from plus or minus DELTA points of the last match with the reference. In this modified

procedure a match is looked for in only the positive direction. This prevents the match from running backward through the reference. That is, the match can not go halfway through the reference template and then progress back to the beginning of the template in case the word happens to be symmetrical. The value of delta chosen determines how much authority the dynamic program will have to expand or compress the reference template. Delta is found experimentally. No analytic method was found to calculate the optimum value for delta.

In the Unconstrained Endpoint Local Minima recursive equation, the index of the first template is used to drive the algorithm. That is, each point of the driving template is taken in order. As a result of only one of the templates driving the algorithm, one should not expect to obtain the same accumulated distance if the driving and reference samples are interchanged. The importance of this difference in distances is that in order to compare accumulated distances, the sample template should be the first or driving template, while the reference template is second. This allows the accumulated distances for different references to be compared against the same scale.

One of the major constaints placed upon dynamic programming algorithm is the treatment of endpoints. Some methods constrain the endpoints of the sample and the reference templates to match exactly. Most researchers using

dynamic programming for speech recognition agree that this is not a reasonable constraint for this application. Due to noise and difficulties in exactly locating the word endpoints, the endpoint position tend to vary from the real endpoints. It is therefore not a reasonable constraint to force points that can not be exactly determined to exactly match each other.

If the method used does not constrain endpoints, it will have to deal with the problem of one of the templates reaching the end before the other. If the driving template reaches the end first, most methods terminate and use the distance accumulated to this point. This method either regards the remaining portion of the reference as noise, or considers the reference endpoint to be misplaced. A problem occurs when the reference template terminates first. In this case the dynamic program does not run through the entire sample, the accumulated distance will not be based upon the correct number of points.

One solution is to continue the method, duplicating the last reference point as many times as necessary until the end of the driving template is reached. This project calls this method, termination with NO INTERPOLATION. A second method, which generally gives better results, is to terminate when the end of the reference is reached, and to scale the resulting distance by (total points in driver/point number of driver at termination). This method

is called termination with INTERPOLATION. In Rabiner's Unconstrained Endpoint Local Minima (UELM) method the accumulated distance at point (n) of the driver is: [5]

$$D_A(n) = D_A(n-1,q) + \min D(n,m)$$

For: $q-delta < m < q+delta$

The total accumulated distance is generated by minimizing the local distance between points but does not guarantee a global minimum path. One important point to note is that since m is constrained to be within plus or minus delta points of q, the match can actually run backwards along the reference. To eliminate this possibility for this project the constraint was changed to $q < m < q+delta$. The UELM algorithm was chosen for this project since it has been shown by Rabiner to give results comparable to other methods while being the least costly as far as computation time. [4] That is, the UELM method which performs only a local minimization gives results comparable to a global minimization but requires much fewer calculations.

3.1.4.1 Thresholds

Thresholds can be used in various ways along with the dynamic programming. In addition to eliminating bad matches and matches that accumulate large errors early, thresholding is very important for the decrease in time that occurs in the dynamic programming routine. The simplest form of

thresholding is to have one maximum value that the distance must be less than in order to be considered a valid match. This method is very useful to eliminate words that are not in the vocabulary.

The next type of threshold is based on the heuristic principle that if a match has a high error value early on, it will be a good candidate to eliminate. This is based on the fact that the error function is a monotonically increasing function. This type of threshold uses a graduated threshold that is lower for early cycles of the dynamic program and increases as the match proceeds. This type of thresholding works best when the accumulated error distance of the desired matches are concave upward, and the undesired matches are concave downward. That is, the desired matches have most of the error occur at the end of the word while the undesired matches have most of the error occur at the beginning of the word. When using a graduated threshold, it is quite possible to have two matches that without thresholding would obtain the same final error value, yet with thresholding, the sample that accumulated its error earlier would be eliminated. The thresholds must be high enough not to eliminate the correct match. By proper selection of thresholds the selection time can be significantly reduced. The thresholds were adjusted by setting them to a value that was twice the accumulated error distance that occured during the match with the correct

template. Setting the thresholds this high prevented the correct template from being eliminated by thresholding. Once the thresholds were raised high enough to prevent the elimination of the correct template, the exact vaule of the threshold was not critical. If there is sufficient time, it is far better to have the thresholds too high than too low as this will prevent to possibility of the correct template from being eliminated by thresholding.

3.1.5 FISHER DISCRIMINANT

The Fisher Linear Discriminate was used on this project as one type of classifier. This classifier determines the equation of a hyperplane that separates the two or more classes of interest. [5] The data sample is then classified by simply determining on which side of the hyperplane the sample falls. In order to determine the dividing hyperplane, the means and the covariance matrices of the classes must be found. The Fisher Linear Discriminant function is a function of the form:

$$H(X) = V^T X + Vo \qquad \begin{array}{ll} IF > 0 & X = w1 \\ IF < 0 & X = w2 \end{array}$$

This function will classify a sample X as belonging to class w1 if $H(X)>0$ and class w2 if $H(X)<0$. The vector V and scalar Vo are found by using the Fisher criterion:

$$f = (n1 - n2)^2 / (\sigma_1^2 + \sigma_2^2)$$

That is, the linear boundary between classes will fall between the means, ni, of the two classes spaced inversely proportional to the variances, σ_i^2, of the classes.

$$V = (.5S1 + .5S2)^{-1} (M1 - M2)$$

where Si is the covariance matrix for class i and Mi is the mean of class i.

$$Vo = \frac{(M2 - M1)^T (.5S1 + .5S2) (\sigma_1^2 M2 + \sigma_2^2 M1)}{\sigma_1^2 + \sigma_2^2}$$

where σ_i^2 is the variance of H(X) for class i and can be calculated from:

$$\sigma_i^2 = V^T Si V$$

The linear discriminant vector V and scalar Vo are derived as follows:

Let $f(n1, n2, \sigma_1^2, \sigma_2^2)$ be any criterion function to be maximized.

Then

$$\frac{\partial f}{\partial V} = \frac{\partial f}{\partial \sigma_1^2} \frac{\partial \sigma_1^2}{\partial V} + \frac{\partial f}{\partial \sigma_2^2} \frac{\partial \sigma_2^2}{\partial V} + \frac{\partial f}{\partial n1} \frac{\partial n1}{\partial V} + \frac{\partial f}{\partial n2} \frac{\partial n2}{\partial V}$$

$$\frac{\partial f}{\partial Vo} = \frac{\partial f}{\partial \sigma_1^2} \frac{\partial \sigma_1^2}{\partial Vo} + \frac{\partial f}{\partial \sigma_2^2} \frac{\partial \sigma_2^2}{\partial Vo} + \frac{\partial f}{\partial n1} \frac{\partial n1}{\partial Vo} + \frac{\partial f}{\partial n2} \frac{\partial n2}{\partial Vo}$$

But

$$ni = V^T Mi + Vo$$

$$\sigma_1^2 = V^T Si V$$

So

$$\partial\sigma_1^2 / \partial V = 2 Si V$$

$$\partial ni/\partial V = Mi$$

$$\partial\sigma_1^2 / Vo = 0$$

$$\partial ni/\partial Vo = 1$$

Substituting

$$2(\partial f/\partial\sigma_1^2) S1 + (\partial f/\partial\sigma_2^2) S2) V = M1 - M2 \; \partial f/\partial n2$$

$$\partial f/\partial n1 + \partial f/\partial n2 = 0$$

Now using the Fisher criterion

$$f = (n1 - n2) / (\sigma_1^2 + \sigma_2^2)$$

$$2((n1 - n2)/(\sigma_1^2 + \sigma_2^2)) (.5S1 + .5S2) V = M1 - M2$$

But the scale factor $2((n1 - n2)/(\sigma_1^2 + \sigma_2^2))$ does not change the slope, so it can be deleted.

$$V = (.5S1 + .5S2)^{-1} (M1 - M2)$$

Since

$$H(X) = V^T X + Vo = 0$$

When

$$X = \frac{\sigma_1^2 M2 + \sigma_2^2 M1}{\sigma_1^2 + \sigma_2^2}$$

And

$$Vo = -V^T X$$

Then

$$Vo = \frac{(M2 - M1)^T (.5S1 + .5S2) \quad (\sigma_1^2 M2 + \sigma_2^2 M1)}{\sigma_1^2 + \sigma_2^2}$$

The Fisher discriminant as defined above is a two class problem. In order to generalize it to a multiclass problem, the Fisher discriminant can be applied to all of the classes by pairs. In order for X to be labeled as class i, the following constraint must be met:

$$V_{ij} X + Vo_{ij} > 0 \quad (j=1,2,...M;\ i<>j)$$

where M is the number of classes. A difficulty that develops with the linear discriminant for a multiple class problem is that it is possible to have regions in space where no consistent classification is possible. These regions, called reject regions, indicate regions where there is no class i in which the above constraint is met. For this reason linear discriminant functions tend to perform poorly for large class problems. In this project the Fisher discriminant is a fairly small class problem for the number of input parameters, so it worked quite well. That is, there were approximately the same number of independent parameters as there were classes to be separated. This meant that the Fisher discriminant could form decision boundaries without creating large reject regions.

3.1.6 STANDARD DEVIATION WEIGHTING

When using several different parameters for pattern recognition, some form of normalization is required to take into account the information content of the different parameters. Normalization is used to take into account that a parameter might be far from the mean, but it should not contribute significantly to the error distance if it has a very large standard deviation. One method discussed by Duda and Hart to normalize data is to subtract the mean of the class and divide by the standard deviation of each component. This method is related, but distinctly different from the weighted distance measures previously discussed, which divide by the variance.

$$\frac{Xi - Mi}{\sigma_i}$$

The total distance to the class i is found by taking the Euclidean distances of the resultant components. The boundaries between classes will remain a straight line, with the slope changed because of dividing by the standard deviation.This weighting will prevent a single component with a large variance that is far from the mean from dominating the distance. [3] This technique of weighting measurements inversely proportional to the variance estimates is a well known technique in Kalman Filtering to

obtain a better estimate of a parameter in the presence of noise.

3.1.7 K NEAREST NEIGHBOR

The K Nearest Neighbor rule is often used as a classification technique when multiple copies of templates are stored. In this project the K Nearest Neighbor rule is used to classify a sample after using the Euclidean or Tchebycheff distance measure when finding the error distance. Instead of simply picking the template that has the lowest error distance, the K Nearest Neighbor routine is passed the error distances of all of the templates, and assigns a sample X to class wi if the majority of the K nearest matches are class wi. If K is fixed and the number of samples is allowed to increase to infinity, then all of the K nearest neighbors will converge to ci. The K Nearest Neighbor rule selects ci if the majority of the K nearest neighbors are ci, with probability:

$$\sum_{i=(K+1)/2}^{K} \binom{K}{i} P(ci/X)^{i} [1-P(ci/X)]^{K-i}$$

This rule is an attempt to estimate the a posteriori probability P(ci/X). One would like to use a large value for K in order to obtain an accurate estimate. A contradictory requirement is that all of the K matches be close to one

class. These two contradictory requirements force K to be a small number compared to the total number of samples. [6]

3.1.8 COMPARISON OF CLASSIFIERS

The K Nearest Neighbor classifier is a non-linear classifier while the Standard Deviation Weighting and Fisher Discriminant are linear classifiers. This non-linearity of the K Nearest Neighbor allows more freedom in the placement of the decision boundary.

The following sample data will be used to show how each of the classifiers forms the decision boundary. Figure 2, shows the data points and the boundaries formed by each classifier.

CLASS 1	CLASS 2
(0, 2)	(4, 2)
(0, 6)	(4,-2)
(-1, 4)	(3, 0)
(1, 4)	(5, 0)

Means of each class are:

M1=(0,4)

M2=(4,0)

The covariance matrices for each class are:

$$S1 = E\,(X-M)(X-M)^T$$

$$S1 = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$$

$$S2 = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$$

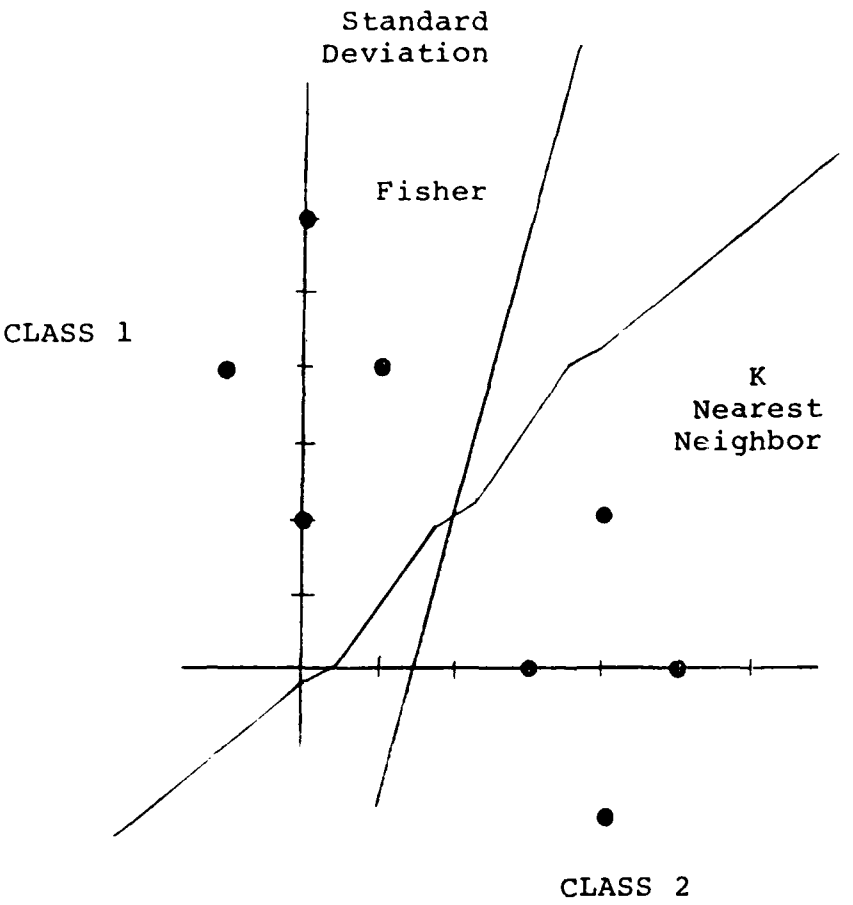


FIGURE 2. COMPARISON OF BOUNDARIES

The standard deviations of each parameter are:

$\sigma x = 1$

$\sigma y = 2$

The Standard Deviation Weighting method which subtracts the mean and divides by the standard deviation for that parameter has the following decision boundary.

$$\left(\frac{x-0}{1}\right)^2 + \left(\frac{y-4}{2}\right)^2 = \left(\frac{x-4}{1}\right)^2 + \left(\frac{y-0}{2}\right)^2$$

which results in the linear decision boundary

$y = 4x - 6$

The Fisher Discriminant function can be found as follows

$$V = (.5S1 + .5S2)^{-1} (M1 - M2)$$

$$V = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}^{-1} \begin{pmatrix} -4 \\ 4 \end{pmatrix}$$

$$V = \begin{pmatrix} -4 \\ 1 \end{pmatrix}$$

$$\sigma_1^2 = V^T S1 V$$

$$\sigma_1^2 = (-4, 1) \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix} \begin{pmatrix} -4 \\ 1 \end{pmatrix}$$

$$\sigma_1^2 = 20$$

$$\sigma_2^2 = 20$$

$$Vo = \frac{(M2 - M1)^T (.5S1 + .5S2)^{-1} (\sigma_2^2 M1 + \sigma_1^2 M2)}{\sigma_1^2 + \sigma_2^2}$$

$$= (4,-4) \begin{pmatrix} 1 & 0 \\ 0 & .25 \end{pmatrix} \begin{pmatrix} 80 \\ 80 \end{pmatrix}$$

$$=6$$

Boundary occurs when

$$V^T X + Vo = 0$$

$$y = 4x - 6$$

This boundary happens to be equal to the Standard Deviation weighting classifier boundary. This occurred because the two parameters were independent. If they were not independent, then the off diagonal terms in the covariance matrix S would not be zero, and the boundaries would have been different.

The K Nearest Neighbor classifier forms a piecewise linear boundary. For this example K=3, since that was the value used latter in the final algorithm. The points where the slope of the resulting boundary change are the locations where a new sample point becomes closer than the previous point. The following seven equations represent the decision boundary.

For $(-\infty,-\infty)$ to $(0,-.25)$

$$(x+1)^2 + (y-4)^2 = (x-4)^2 + (y+2)^2$$

$$y = 5/6\ x - .25$$

For $(0,-.25)$ to $(.5,0)$

$$(x-1)^2 + (y-4)^2 = (x-4)^2 + (y+2)^2$$

$$y = .5\ x - .25$$

For $(.5,0)$ to $(1.75,1.88)$

$$(x-1)^2 + (y-4)^2 = (x-4)^2 + (y-2)^2$$

$$y = 1.5\ x - .75$$

For (1.75,1.88) to (2.25,2.13)

$$(x-1)^2 + (y-4)^2 = (x-3)^2 + (y-0)^2$$

$$y = .5\,x + 1$$

For (2.25,2.13) to (3.5,4)

$$(x-0)^2 + (y-2)^2 = (x-3)^2 + (y-0)^2$$

$$y = 1.5\,x - 1.25$$

For (3.5,4) to (4,4.25)

$$(x-0)^2 + (y-6)^2 = (x-3)^2 + (y-0)^2$$

$$y = .5\,x + 2.25$$

For (4,4.25) to (∞,∞)

$$(x-0)^2 + (y-6)^2 = (x-5)^2 + (y-0)^2$$

$$y = 5/6\,x + 11/12$$

Figure 2 shows the 3 resulting decision boundaries plotted with the sample data. The non-linearity of the K Nearest Neighbor boundary allows more flexibility in the decision boundary placement. Notice that the Standard Deviation Weighting and Fisher Discriminant classifiers require statistics for the data classes to be estimated while the K Nearest Neighbor does not. Due to the above reasons, the K Nearest Neighbor classifier was chosen for this project.

3.2 ALGORITHMS

Combining several of the above techniques produced the Non-Uniform and the Uniform algorithms that were used on

this project. The major difference between the two algorithms is in the determination of when time slices should be taken.

When trying to find a match between two signals there are three basic variations to which the algorithm should remain invariant. The first common variation is that of identical signals which differ in amplitude only. With this amplitude variation one of the signals is simply a larger version of the other. In speech systems this difference in amplitude can occur because the word is spoken softer or louder. In order to compensate for the variations in volume of the spoken word, the pre-processor in this system contains an automatic gain controlled amplifier. The AGC amplifier attempts to maintain a constant amplitude regardless of how loud or soft the word was spoken.

The second variation that can occur is that of a time shift. Since this system starts to sample when a set threshold is exceeded, a slight noise or variation in amplitude can change the point at which the sampling starts. By performing a convolution with an edge operator, the points of transition can be found. The convolution is invariant for a time shift and therefore finds the desired points of transition.

The third problem to be dealt with is that of comparing signals with different numbers of samples. The signals that are sampled can have different numbers of points since the

signals are sampled nonuniformly at points where the composite gradient signal indicates stationarity. Noise in the original signals can cause a different number of samples to be taken. In order to match signals containing a different number of points, the system uses dynamic programming. The technique used is known as unconstrained endpoint local minima. The way that the algorithm is set up in this system is that the test utterance is compared to each of the references, with the test utterance driving the procedure. The first point of the test is compared with the first, second, and third point of the reference. The best match is then found using a weighted Euclidean distance. The second point of the test is then compared with the best point found in the preceding match plus the next two points of the reference. Each point of the test utterance is compared similarly until the end of the test utterance. This method allows the program to find a low distance match of signals with different numbers of points.

3.2.1 NON-UNIFORM ALGORITHM

This algorithm attempts to sample the parameters non-uniformly at the points where the signals are wide sense stationary. The goal of this algorithm, developed by the author, was to make use of a local operator to find the edges in the envelope detected waveforms so that the number of time slices required could be reduced. The following is a

summary of the algorithm followed by an explanation for each of the steps.

1. Take 128 samples of each of 8 envelope detected signals at 10 ms intervals. The sampling is started when a threshold is exceeded.

2. Run the local operator -1 -2 -3 -2 -1 0 1 2 3 2 1 across each of the 8 sampled signals.

3. Form a composite signal from the average of the absolute value of the 8 gradient signals.

4. Select slices from the original sampled data at places where the composite gradient is close to zero.

5. Use the Unconstrained Endpoint Local Minima (UELM) dynamic programming method to find the weighted Euclidean distance to each of the reference patterns. Use thresholds to reduce computation by terminating unpromising matches.

6. Use K Nearest Neighbor with k=3 to find the most likely match.

Step 1 which samples at 10 ms intervals is a compromise between required storage and desired information. The shortest English phoneme according to Votrax, a manufacturer of voice synthesis hardware, is approximately 47 ms long. By sampling at 10 ms intervals we can have several samples per phoneme. The result is that the sampling window is 1.28 seconds long. Figure 3 shows a typical pattern of the 8 channels for the word 'seven'.

Step 2 which runs a local operator across the signals is, in essence, a convolution looking for ramps in the signals. Due to the lowpass nature of the envelope detected signals, step transitions will not occur. The size of the operator is matched to the size of transitions that occur. By correctly matching the operator size, noise can be smoothed out while at the same time accentuating the desired ramp transitions. The local operator is performing the function of a matched filter which indicates when a desired waveform occurs.

Step 3 forms a composite signal that represents steady state conditions of the original data. A steady state condition is indicated by a value of the composite signal near midscale. Figure 4 shows the individual signal with the local operator applied on the lower 7 traces and with the composite signal on the upper trace.

Step 4 selects the appropriate time slices from the 8 envelope detected signals for matching. These are time

FIGURE 3. WORD 'SEVEN'

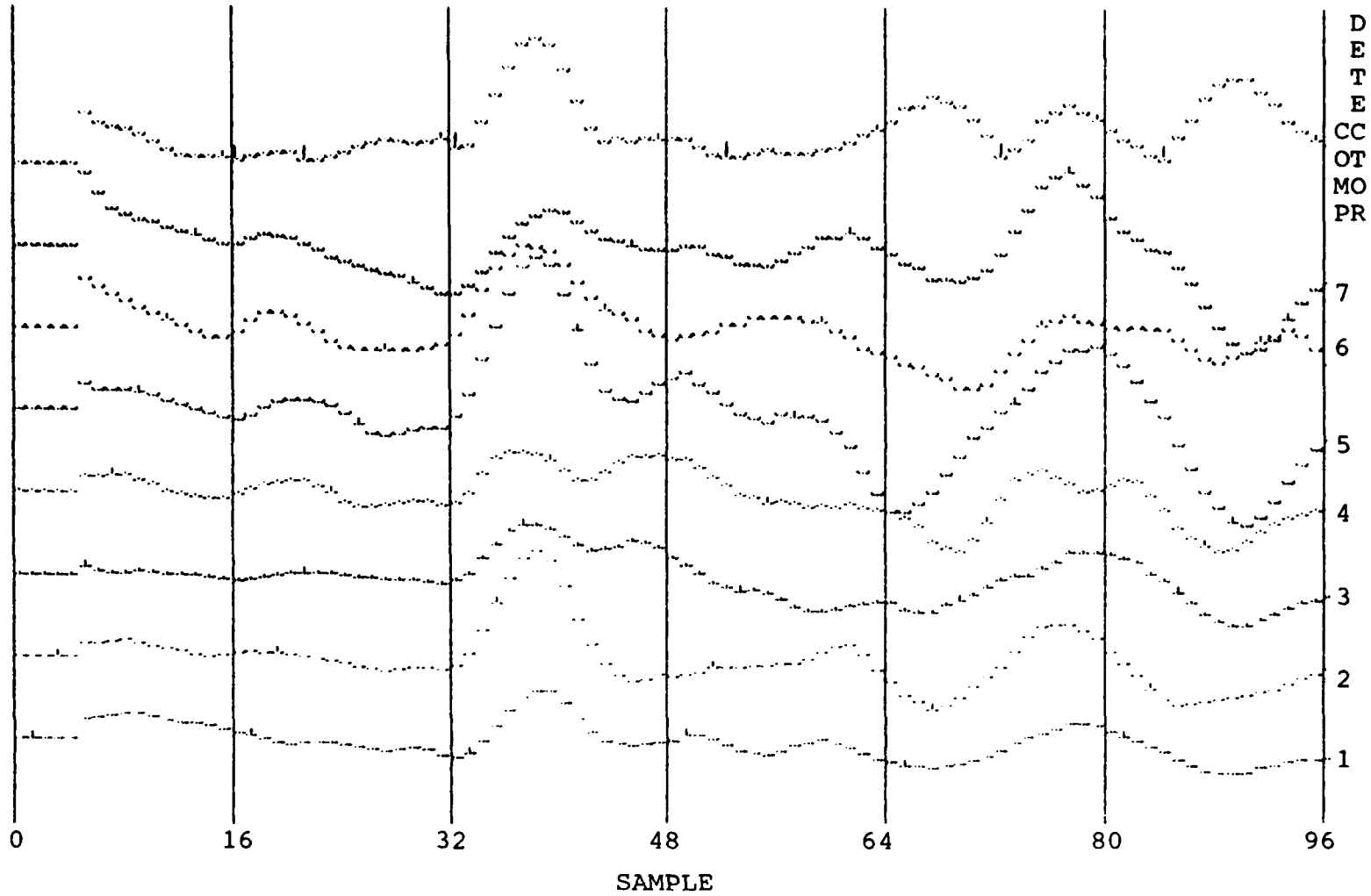


FIGURE 4. WORD 'SEVEN' AFTER LOCAL OPERATOR

slices where the signals are wide sense stationary. This is an attempt to insure that a time slice is taken through each of the continuant phonemes.

Step 5 uses an unconstrained local minimum dynamic programming technique to perform a warping in order to permit the comparison of samples of different lengths.

Step 6 uses the K Nearest Neighbor classifier so that more than one reference pattern can be compared. The K Nearest Neighbor classifier was selected over the Fisher Linear Discriminant and the Standard Deviation Weighting classifiers, because the K Nearest Neighbor allows a non-linear decision boundary which can better approximate the optimum decision boundary.

3.2.2 UNIFORM ALGORITHM

The Uniform algorithm is basically the same as the Non-Uniform algorithm with the exception that samples are taken at fixed time increments, regardless of whether the signals are constant or not.

1. Take 128 samples of each of 8 envelope detected signals at 10 ms intervals. The sampling is started when a threshold is exceeded.

2. Subsample the data to obtain the desired number of samples.

3. Use the Unconstrained Endpoint Local Minima (UELM) dynamic programming method to find the weighted Tchebycheff distance to each of the reference patterns. Use thresholds to reduce computation by terminating unpromising matches.

4. Use K Nearest Neighbor with k=3 to find the most likely match.

One inherent advantage of the Uniform algorithm is that for the same number of time slices, it is shorter. Therefore, it can be performed quicker with less computing power. This advantage only applies for the same number of time slices. Normally one would expect to need more time slices when using the Uniform algorithm in order to be assured of having a time slice through each of the continuant phonemes. The reason is that phonemes have different lengths. In order to be assured of obtaining a time slice through each continuant phoneme, it is necessary in the uniform sampling case to sample at least as often as the shortest phoneme of interest. This entire argument is based on the assumption that the continuant phonemes contain the information of interest.

3.3 VARIABLE DIMENSION STATISTICS

The varying rates at which words are spoken cause a significant problem when trying to calculate the statistics needed for the different recognition methods. The normal definitions of averages and variance do not apply since the dimensionality of the input parameters differs for different samples of the same word. The dimensionality varies because a particular time slice of one word does not represent the same information as the same time slice through a second utterance of the same word. It therefore became necessary to define both what will be considered to be an average and what is the measure of variance. Since the non-uniform samples are handled by the dynamic programming in the matching algorithm, the averaging should also be able to be performed by using dynamic programming. In order to find the average, the routine starts with one sample of the word and runs the dynamic programming to find the best fit with the sample to be averaged. This type of averaging is done with each successive word to be averaged. This form of averaging is highly dependent on two starting conditions, the sample that drives the dynamic programming routine and the distance measure. Since one of the reasons for performing the average is so that the variances can be calculated for use with the weighting of the Euclidean distance, and the averaging uses this distance measure to perform the average, the procedure must be iterative. The variances are calculated in a

similar manner, using the averages generated by the averaging routine to drive the dynamic programming. The variance is calculated for each of the eight envelope detected signals. The reciprocal of these variances are then used as the weightings when finding the Euclidean distance between points. Thus the two procedures of finding the average and finding the variance are inseparably interlocked.

3.4 SYSTEM

The initial computer system consisted of a microprocesser system based on a Z-80. This system was specifically designed by the author for ease of use with hardware experiments. The heart of the system was a Z-80 CPU running at 4MHz with one wait state. The memory consisted of 64K of Read/Write memory with the upper 2K shadowed by a PROM with monitor routines. On line storage included two 8 inch single density floppy disk drives. The system had an analog board capable of 8 channels of analog input in a range of 0-5 volts, feeding an 8 bit analog to digital converter. The data analysis and verification was done on an IBM 3033 with the Michigan Terminal System operating system. This system was used because of its ability to access and store large data bases quickly. This system also had a large library of programs that were useful in the analysis. This system had a dial-up capability that allowed the target

system to transfer data over a 1200 baud modem to the IBM 3033.

3.4.1 HARDWARE

The external equipment consisting of amplifiers, filters and envelope detectors was designed and built specifically for this project by the author. The output of the microphone is fed into an Automatic Gain Control (AGC) amplifier. The output of the AGC amplifier is fed to seven bandpass filters. The outputs of the seven bandpass filters are buffered, envelope detected, and then buffered again. This results in eight envelope detected signals, seven from the bandpass filters and one unfiltered signal. The bandpass filters are spaced on a logarithmic scale and range from approximately 300 to 3000 Hertz. This range from 300 to 3000 Hz was chosen since this is the range of a typical voice communication link such as the telephone. A block diagram of the external hardware is shown in figure 5. The output of this external equipment is connected to the 8 input channels on the analog board. The microphone used was an electret microphone, Realistic #33-1050. Figure 6 shows the automatic gain control (AGC) amplifier used on the input. The variable gain element was an LM370. The output of the AGC circuit had a level of approximately 5 volts peak to peak. The output of the AGC was fed to the filters. The filters where Reticon R5620 universal active filters set up in a bandpass

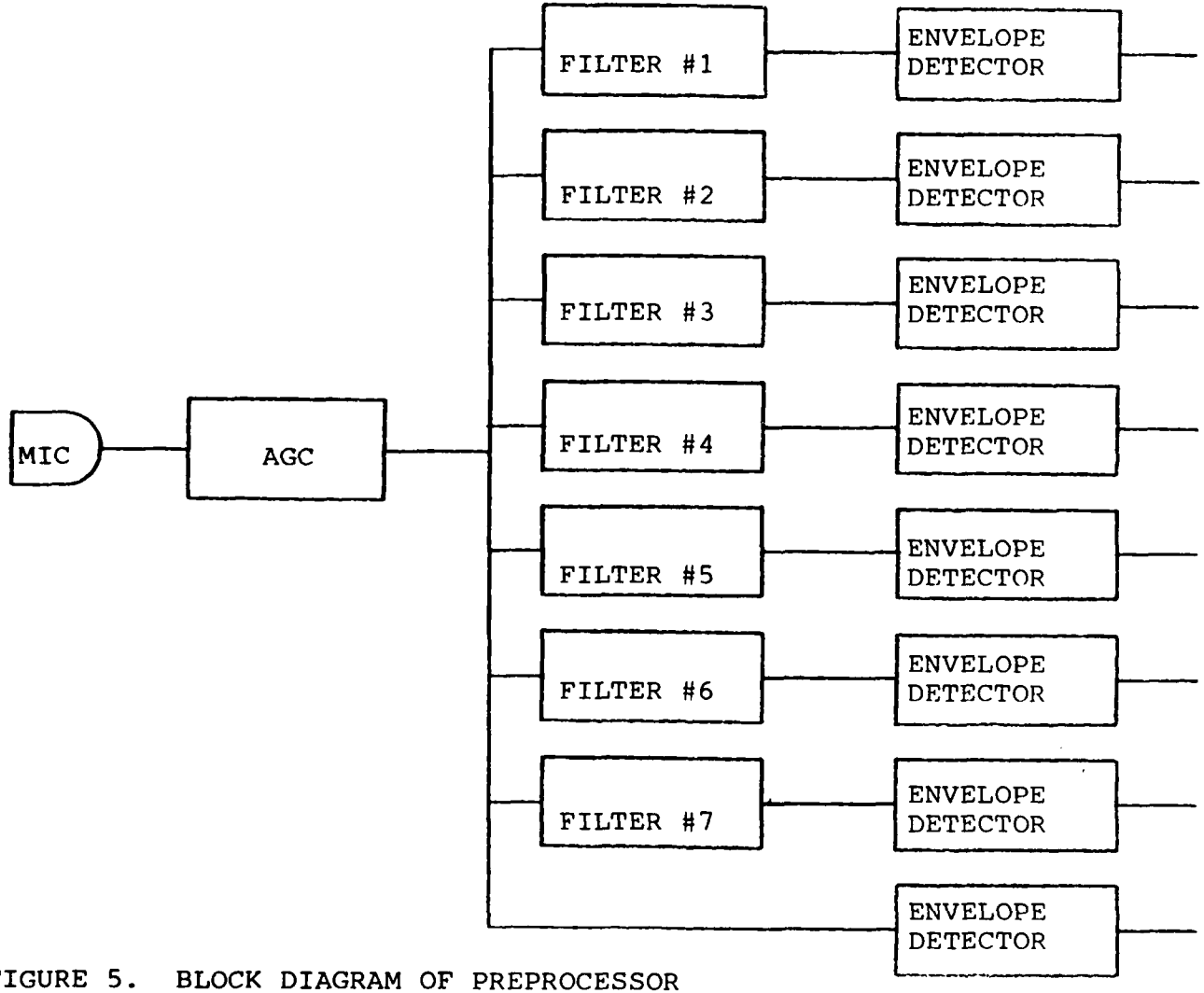


FIGURE 5. BLOCK DIAGRAM OF PREPROCESSOR

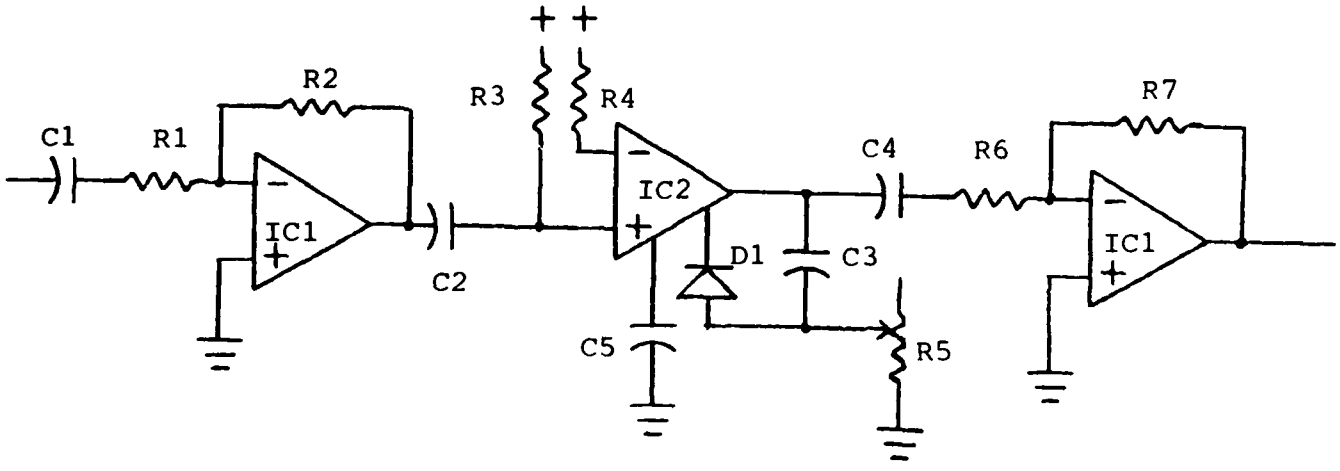


FIGURE 6. AGC SCHEMATIC

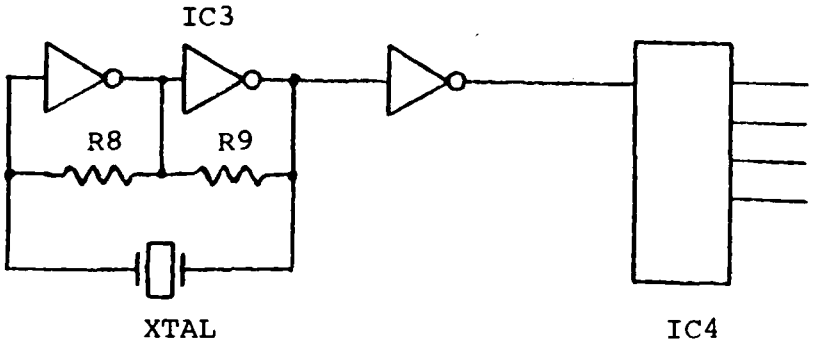


FIGURE 7. CLOCK GENERATOR CIRCUIT

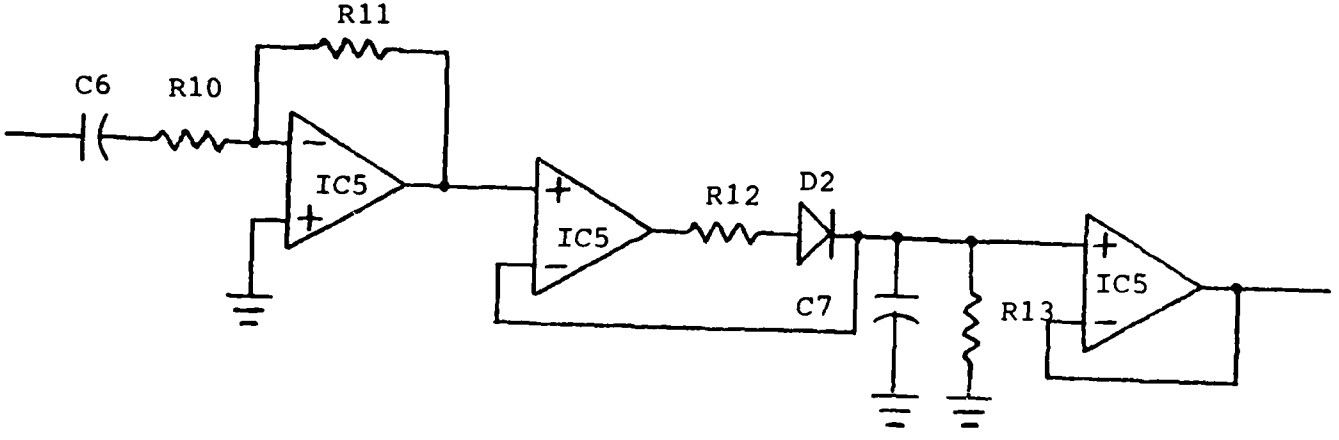


FIGURE 8. ENVELOPE DETECTOR CIRCUIT

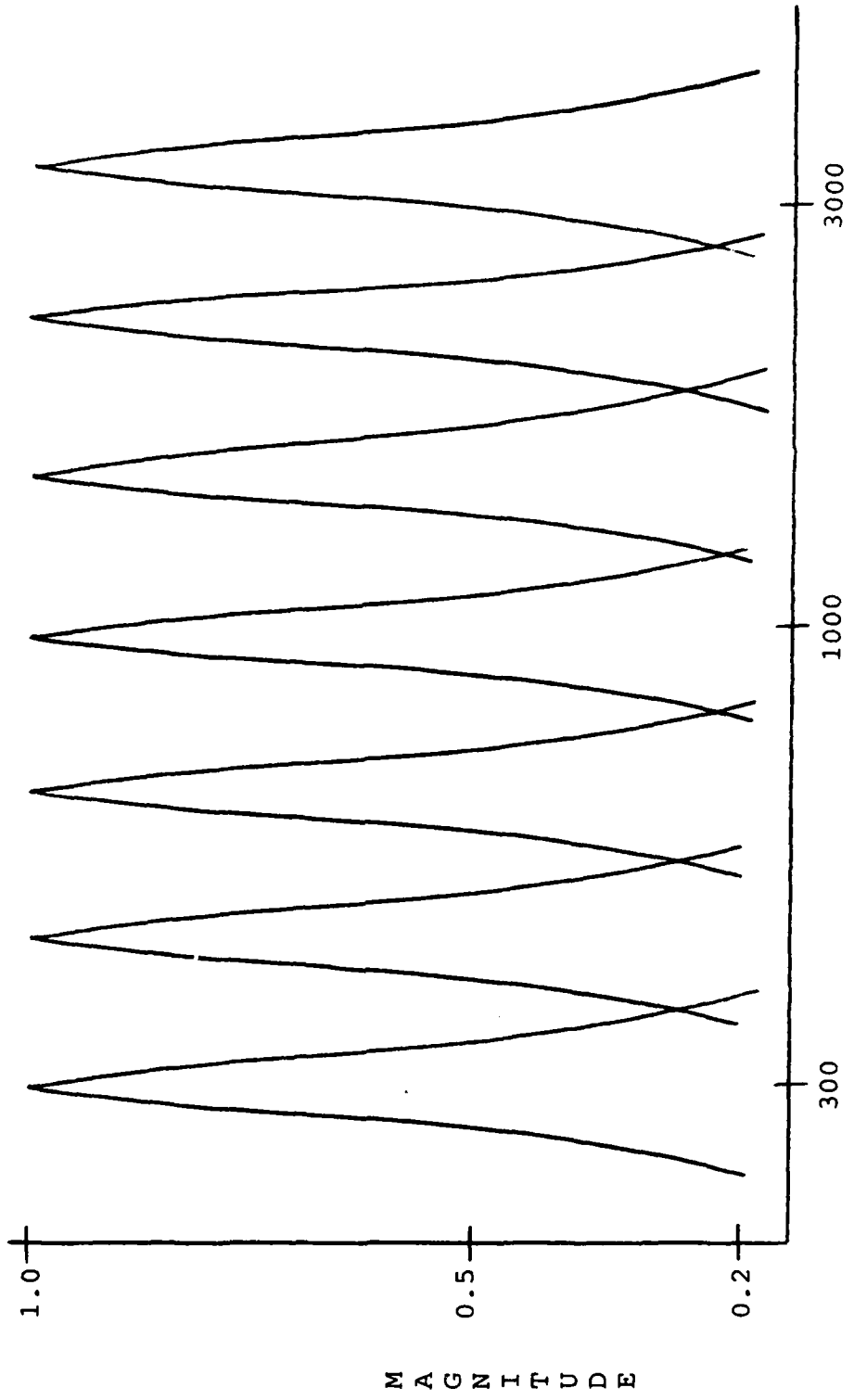


FIGURE 9. FILTER RESPONSES

configuration. [7] These filters are second order switched capacitor networks. The frequencies of the filters are dependent upon an external clock. The clock generator circuit is shown in figure 7. The external frequencies and center frequencies of the bandpass filters are shown in the following table.

FILTER	CENTER FREQUENCY	EXTERNAL CLOCK
1	305	31.25KHz
2	447	62.5 KHz
3	653	125.0 KHz
4	977	125.0 KHz
5	1495	250.0 KHz
6	2236	250.0 KHz
7	3342	250.0 KHz

TABLE 4. FILTER FREQUENCIES

All of the filters were programmed for a Q of 10. A plot of the filter responses is shown in figure 8. The output of the filters are fed into envelope detectors shown in figure 9. The envelope detectors had a time constant of 33 ms. The signal after the detector is buffered by a unity gain amplifier.

3.4.2 SOFTWARE

The programs in appendix D are the final programs written by the author. Portions of the programs were written in PL/M and assembly language.

TABLE 5. SOFTWARE CALLING TREE

The program INIT is used to obtain the templates that will be used by the program EAR to recognize a word. The procedure DIF3 is called by EAR and performs the dynamic programming. The procedure SAMP is called by both EAR and INIT and is used to take 128 samples of the 8 envelope detected waveforms at 10 ms intervals. The recognition program takes 3-4 seconds to respond to a spoken word, and requires under 8K of memory for the templates and the program. The program does not require any special computer architecture, so should be easily adapted to run on other systems.

4.0 COST

The additional hardware required for this project can be built for under $150. A large factor in the low cost of this hardware was the availability of a new Universal Active Filter from Reticon, the R5620. This filter costs under $7 and requires no external precision components. This system is low cost when compared to the $1K-$5K cost of a commercial system made by Votan, Lear Siegler, or Interstate

Electronics. The $1K-$5K cost of a commercial system is as much as, or more than, the cost of the computer systems that they would be attached to. This large investment has deterred most users from adding a voice input capability to their systems.

5.0 TEST AND EVALUATION

Once the hardware and software were designed and implemented, the next step was to evaluate the system performance. The testing was performed in a stepwise manner in order to optimize portions of the algorithm.

5.1 DATA BASE

In order to analyze the performance of the non-uniform algorithm, a data base was gathered. The data base consisted of 10 samples of each of the ten words zero through nine, from nine different speakers. This data base was gathered using the target computer. The data was then transferred to the IBM 3033 for analysis.

The first five speakers were adult males between the ages of 20 and 40. The second four speakers were adult females between the ages of 18 and 45. The data was gathered directly on the system without any means of intermediate recording. No intermediate recording was done of the data to prevent the recording process from adding noise or distortion to the desired signal. The data gathering was

performed in a relatively quiet, furnished apartment. The main noise source present was the air noise from the system cooling fans.

In order to compare the performance of the Non-Uniform to the Uniform algorithm, a second data base was gathered. All of the tests in sections 5.2 and 5.2.1 were done using data in the first data base while all of the tests in section 5.2.2 were done using data from the second data base. It was necessary to gather a second data base since the original data base consisted of the time slices after the non-uniform sampling. This second data base consisted of both the non-uniform time slices and the uniform time slices. Both sampling methods were used on the same utterance so that a valid comparison could be made of the two sampling techniques. While this was not ideal, it was necessary to store only the time slices used in order to limit the amount of required storage. The second data base consisted of five speakers, one adult female and four adult males. These speakers were a subset of the individuals used in the first data base, recorded under the same conditions.

When analyzing the performance of the algorithms, the data was used in two different tests. First, the data from a single speaker was used. When using a single speaker, the first five utterances of each of the ten words were averaged to form that speakers template for that word. The second five utterances of each of the words were not used so

that they could be kept as a control group. Each of the utterances from that speaker was compared against the ten templates. The second way the data was used was to compare data from five speakers at a time. In this case, each utterance was compared against 50 templates, the ten words from each of the five speakers.

A confusion matrix is a matrix used to represent the performance of a system. Each row in the matrix represents the actual word spoken. Each column in the matrix represents a word that the system picked as the answer. The entries in a row represent the number of times that the system picked the template, corresponding to the column number, as the best match.

Confusion matrices were generated for the five speaker tests which had the words zero thru nine, in order, repeated five times, once per speaker, along each edge. These confusion matrices show what word was recognized for each of the actual spoken words.

5.2 RESULTS

The first area to be investigated was that of the weighting coefficients. The first weighting to be used was 1 2 2 4 2 2 1 8. These are the weighting coefficient w1 through w8, in order, used with the weighted distance measures. This set of weightings was experimentally determined in the early stages of development to be a

reasonable starting point. The starting point did not really matter since the procedure of finding an average and calculating the variances was repeated until the weighting coefficients were inversely proportional to the calculated variances. The goal of the weighting was to weight each parameter inversely to its variance. The following table lists the various weightings used. The final weighting settled upon was 4 4 1 1 2 2 2 1 which resulted in 74% correct recognition of the training group and 74% correct recognition of the control group. With this weighting of 4 4 1 1 2 2 2 1 the desired weighting inversely proportional to the variance was obtained. Since the variance was dependent upon the average which was dependent on the weighting coefficients, an iterative approach was necessary. The iterative procedure consisted of using one set of weightings to find an average, using this average to calculate the new variances, and using these new variances to set the weighting coefficients. This procedure was repeated until the weightings converged to be inversely proportional to the calculated variances. This problem of interdependence was discussed in section 3.3.

WEIGHTING	TRAIN	CONTROL
1 2 2 4 2 2 1 8	70%	
2 2 1 2 1 2 2 4	70%	63%
4 2 1 1 1 2 2 4	68%	
4 4 1 1 2 2 2 1	74%	74%

TABLE 6. EFFECTS OF COEFFICIENT WEIGHTING

As can be seen, the algorithm is not extremely sensitive to the weighting coefficients. However, the last set of weightings, which was approximately inverse to the calculated variance, did obtain the best results. The weightings were purposely kept as integer powers of 2 in order to facilitate programming on the target computer.

An attempt was made to use different weightings for different reference words. This resulted in a reduction of the correct recognition rate to 63%-64%. This was not totally unexpected since the errors being compared are measured with rulers of different scales when different weightings are used.

The next area to be investigated was that of thresholding. Thresholds were determined from the training data that would allow each of the training words to progress through a match with the proper reference pattern without reaching the threshold. These thresholds were then used in the algorithm. The result was a 79% correct recognition from the training set but a decrease to 66% correct recognition from the control group. The rapid decline in the control

group was caused by the thresholds being too low, which caused the correct template to be eliminated. When the thresholds were raised to twice their lowest value, the percentage of correct recognition returned to its former value. This shows that although it is possible to use low thresholds to change the outcome, the values low enough to do this are quite critical. The real value of thresholding for this project is the reduction in time for the algorithm to run by eliminating matches that have very high errors early. The results of how well the data clustered for each speaker was determined by using an average only from that speakers words. The first set of data is without termination INTERPOLATION as discussed in section 3.1.4

SPEAKER	TRAIN	CONTROL
1	100%	92%
2	72%	68%
3	90%	74%
4	82%	78%
5	98%	94%
6	86%	
7	90%	
8	94%	
9	90%	

TABLE 7. RESPONSE WITHOUT INTERPOLATION

As is quite evident from the above, speaker 2 and 4 did not cluster very well. This could also explain why the overall recognition rate for the five speakers together was low.

The termination procedure was modified for the dynamic programming algorithm. Previously the algorithm would continue for the full length of the test utterance regardless of whether the end of the reference pattern was reached. The result was that if the test pattern is longer than the reference, then the entire difference will be added as error. With the modification, the procedure stops when the end of the reference is reached. At this point, the error is multiplied by the length of the test utterance and divided by the point where the dynamic program terminated. This results in a linear interpolation of the error at termination.

SPEAKER	TRAIN	CONTROL
1	100%	94%
2	88%	78%
3	94%	82%
4	78%	76%
5	98%	98%

TABLE 8. RESPONSE WITH INTERPOLATION

When compared with the unmodified procedure, the results are a higher or equal percentage of correct recognition in all cases except one. The decrease in correct recognition that occurred when the interpolation is not used is caused by the mismatched endpoints contributing too high a percentage of the error. The results of this test show that the method of linear interpolation as advocated by Rabiner does provide

better results than dynamic programming without interpolation.

The next test run was to use one template for each word per speaker. In this case, this meant five speakers at ten words each for a total of fifty reference templates. This test allows the comparison of each of the three speech recognition problems, Word recognition, Speaker recognition, and Word-Speaker recognition. The following table summarizes the results.

	TRAIN	CONTROL
WORD-SPEAKER	72%	60%
WORD	83%	78%
SPEAKER	76%	65%

TABLE 9. 50 CLASS PROBLEM WITHOUT INTERPOLATION

This test confirmed that Word-Speaker recognition is the hardest type of speech recognition to perform. In order to perform Word-Speaker recognition, the system must perform both Word and Speaker recognition. It would therefore be unreasonable to expect higher recognition rates on the Word-Speaker problem than on the lowest of the Word or Speaker recognition problems.

The 50 different error distances were rank ordered. The second best match, second lowest error distance, was examined to see if there was a good likelihood that some further processing of the top two results could increase the

percentage of correct recognition. The confusion matrix was done for the second best match, with the following results.

	TRAIN
WORD-SPEAKER	15%
WORD	59%
SPEAKER	26%

TABLE 10. 50 CLASS PROBLEM WITHOUT INTERPOLATION, SECOND BEST

The combining of the results gives 87% correct word-speaker recognition in the first two answers out of 50 possible choices. The combined recognition rate of 87% correct in the top two responses out of 50 possible responses gave an indication that other classifiers should be examined. These classifiers are the subject of the tests shown in section 5.2.1.

The above confusion matrices were redone with two changes. The dynamic programming routine termination procedure was modified as discussed above, and the number of time slices used as a ninth parameter. Figure 20 and 21 show the resulting confusion matrices for the training and control groups. The results were summarized as follows.

	TRAIN	CONTROL
WORD-SPEAKER	81%	66%
WORD	90%	81%
SPEAKER	86%	70%

TABLE 11. 50 CLASS PROBLEM WITH INTERPOLATION

The second best choice was also run for the training set.

	TRAIN
WORD-SPEAKER	14%
WORD	61%
SPEAKER	25%

TABLE 12. 50 CLASS PROBLEM WITH INTERPOLATION, SECOND BEST

Combining the best two answers on the training data results in 95% correct word-speaker recognition in the top two answers. Again there is confirmation of the previous results that linear interpolation is the superior termination method, and that the second best choice out of 50 contains a significant portion of the correct responses.

Changing the speakers and using data from speakers 1,3,5,8,and 9 to eliminate the suspect data from speakers 2 and 4 gave the following results:

	TRAIN	CONTROL
WORD-SPEAKER	89%	71%
WORD	95%	80%
SPEAKER	90%	78%

TABLE 13. 50 CLASS PROBLEM WITH INTERPOLATION, SPEAKERS 1,3,5,8,9

As can be seen from comparing the control WORD recognition results, the change of speakers was essentially insignificant.

5.2.1 CLASSIFIERS

Since the 50 class statistics showed that very high percentage rates of correct recognition occurred in the top two answers, it was decided to see if some other form of classifier could improve the results. Different classifiers were tried on the 10 class problems for each speaker. The error distances for each of the words was used as the input parameters. Three different methods of classification were tried, K Nearest Neighbor (original method), Fisher Linear Discriminant, and Standard Deviation Weighting. The following table summarizes the results.

	K Nearest Neighbor		Fisher		Standard Deviation	
Speaker	Train	Control	Train	Control	Train	Control
1	100%	94%	94%	82%	100%	90%
2	88%	78%	82%	58%	86%	52%
3	94%	82%	96%	66%	100%	76%
4	78%	76%	86%	72%	92%	70%
5	98%	98%	100%	78%	98%	82%

TABLE 14. EFFECTS OF CLASSIFIERS

As can be seen from the above table, the K Nearest Neighbor method provided the best results. The Fisher and Standard Deviation classifiers did not perform as well as the K Nearest Neighbor classifier. This was the expected result,

as the Fisher and Standard Deviation classifiers have linear boundaries between classes. The non-linear boundaries of the K Nearest Neighbor classifier allows a closer approximation to the intersections of the probability density functions when the classes have equal a priori probabilities. Using the intersections of probability density functions is the well known technique of a Bayes Classifier which provides the optimum decision boundary.

5.2.2 NON-UNIFORM VS UNIFORM

At this point it was determined that the non-uniform algorithm had been heuristically optimized as well as possible. The next step was to compare the change in performance when the sampling method was changed from non-uniform to uniform sampling. Five methods were compared, non-uniform, uniform with 8 slices, uniform with 16 slices, uniform with 16 slices with a delta of 6, and uniform with 8 slices with the dynamic programing delta increased from 3 to 5. Changing the delta of the dynamic program changes the amount of authority that the dynamic program had to expand or contract the reference template.

SPEAKER	NON-UNIFORM	UNIFORM 8 SLICES	UNIFORM 16 SLICES	UNIFORM 16 SLICES DELTA 6	UNIFORM 8 SLICES DELTA 5
1 TRAIN	94%	98%	98%	98%	92%
1 CONTROL	72%	98%	96%	98%	86%
2 TRAIN	96%	100%	98%	98%	86%
2 CONTROL	76%	98%	70%	100%	70%
3 TRAIN	86%	94%	90%	92%	84%
3 CONTROL	80%	90%	84%	88%	88%
4 TRAIN	92%	100%	90%	90%	94%
4 CONTROL	82%	96%	84%	90%	90%
5 TRAIN	98%	100%	100%	98%	90%
5 CONTROL	96%	98%	98%	98%	94%

TABLE 15. COMPARISON OF UNIFORM VS NON-UNIFORM SAMPLING

The uniform 8 slice algorithm with the standard delta of 3 was found to be superior to any of the other methods. This is especially apparent when looking at the control group results. This combination of 8 slices with a delta of 3 provided the dynamic program enough information and sufficient authority to perform the warping. This test showed that increasing the number of time slices does not necessarily mean better performance. The reason that increasing the number of time slices does not necessarily increase performance is that the allowable dynamic program paths change. That is, paths that previously existed are no longer permitted, and new paths that did not previously exist are now allowed. The data also demonstrated that the experimentally determined parameter delta, which controls the authority of the dynamic program, had an experimentally determined optimal value of 3. Increasing or decreasing the

value of delta caused the results to decline. The Non-uniform algorithm did not perform as well as expected. This was most likely caused by the improper selection of the non-uniform samples. Other schemes of picking time slices should be considered.

The next test was to look at the differences between the Euclidean and Tchebycheff distance measures. The Uniform, Non-uniform algorithms, and a method without dynamic programming were run with both distance measures in order to compare the results.

	NON-UNIFORM		UNIFORM 8 SLICES		NO DYNAMIC PROGRAMMING	
	EUCLID	TCHEB	EUCLID	TCHEB	EUCLID	TCHEB
SPEAKER						
1 TRAIN	94%	96%	90%	98%	64%	72%
1 CONTROL	72%	80%	96%	98%	54%	60%
2 TRAIN	96%	98%	98%	100%	24%	38%
2 CONTROL	76%	76%	92%	98%	36%	38%
3 TRAIN	86%	84%	84%	94%	54%	62%
3 CONTROL	80%	82%	78%	90%	42%	54%
4 TRAIN	92%	96%	96%	100%	58%	64%
4 CONTROL	82%	80%	90%	96%	64%	70%
5 TRAIN	98%	96%	98%	100%	72%	80%
5 CONTROL	96%	96%	96%	98%	64%	76%

TABLE 16. COMPARISON OF EUCLIDEAN VS TCHEBYCHEFF

As can be seen from the above results, the best method still is the Uniform algorithm run with 8 time slices and the Tchebycheff distance measure. This test dramatically showed the effects of the dynamic programming. Without some form of

time warping the results fell off drastically. This drastic decline in correct recognition was another confirmation of the inherent time variability of spoken words.

The last major test run was to run all five speakers utterances against the templates of each of the speakers words. The resulting confusion matrices are shown in appendix C, and tabulated below.

	NON-UNIFORM		UNIFORM 8 SLICES	
	TRAIN	CONTROL	TRAIN	CONTROL
WORD-SPEAKER	86%	63%	96%	87%
WORD	91%	86%	98%	96%
SPEAKER	90%	70%	97%	90%

TABLE 17. 50 CLASS PROBLEM, NON-UNIFORM VS UNIFORM

The Uniform algorithm with 8 time slices performed very well for the intended word recognition problem giving 98% correct responses for the training data and 96% correct for the control data. A significant observed from these results is the indication that the templates used were valid. That is, the same templates performed almost as well on the control data as the training data. This test showed that the final Uniform algorithm with 8 time slices and a dynamic programming delta of 3 was able to give comparable results to the word recognition systems listed in section 2.0.

6.0 DISCUSSION AND CONCLUSIONS

The first part of this project after the hardware was verified to be operational was to determine the weighting factors to be used with the distance measures. An optimum weighting was found, and it was also found that the algorithm had a low sensitivity to changes in the weighting coefficients. The next area of investigation was that of the termination procedure for the dynamic programming section. It was found that scaling by the stopping point as discussed by Rabiner to be the superior method. Different classifiers were used with the minimum distance classifier performing the best. At this point it was determined that it would take a major change to the algorithm to improve the results.

The changes made to the Non-Uniform algorithm resulted in the Uniform algorithm. There were now three parameters that could be changed in order to optimize the algorithm. These three parameters were the number of time slices used, the value of delta in the dynamic programming, and the distance measure. No analytical method was discovered to optimize the the value of delta. The value of delta was experimentally determined by comparing the effects of different values. In comparing all of the tests run, it was shown that the Uniform algorithm using weighting coefficients of 4 4 1 1 2 2 2 1, using 8 time slices, a delta of 3, and the Tchebycheff distance measure, performed the best for word recognition. This algorithm was arrived at

by optimizing each part of the algorithm separately.

The non-uniform and uniform algorithms were compared with a straight forward method that did not use dynamic programming or weighting coefficients. Without dynamic programming or weighting coefficients the results fell off dramatically. The reduction in correct recognition was due to the varying rates at which words are spoken. Without some form of compensation for the varying rates, poor results can be expected. This comparison showed that the dynamic programming was absolutely essential. Unfortunately the non-uniform method did not perform as well as expected. Methods of picking non-uniform time slices warrant further investigation.

An important sidelight to this project was the discovery of a method for finding the statistics of data that had a varying number of parameters. The method was to use the dynamic programing to calculate the statistics. The results from the non-uniform algorithm were good enough to show that this method of using dynamic programming to find statistics provided a reasonable template, average, for the varying parameters.

The proposed algorithm was implemented on the target system. The program fulfilled the project objectives, with the only drawback being that the program takes 3-4 seconds to respond. This program could easily be sped up on a system that had multiprocessing capability. However, most

microcomputers available today do not have multiple processors. Except for being a little slower than desired, the programs performed very well and can easily be adapted for use on any of the present day 8 bit microprocessors. Appendix D shows the programs that were written. The program INIT stores samples of the speakers voice to be used as the templates. The program EAR performs the actual recognition.

The low cost, under $150, makes this hardware configuration very appealing. Since the filters used require no external precision components, only a clock, set up was very easy. There was no tuning required of the filters. The center frequency was dependent on the digital inputs and the external clock. This meant that no precision equipment was needed to align the filters. The programs and hardware described in this paper can easily be implemented to produce a practical, cost constrained speech recognition system.

REFERENCES

1. Lea, Wayne a., Trends in Speech Recognition, Prentice Hall, New Jersey, 1980. p63-65.

2. Rabiner, L. R. and R. W. Schafer, Digital Processing of Speech Signals, Prentice Hall, New Jersey, 1978, p38-88.

3. Duda, Richard O. and Peter E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973, p213-217.

4. Rabiner, Lawrence R., Aaron E. Rosenberg and Stephen E. Levinson, 'Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition', IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol ASSP-26 No 6 DEC 1978, p575-582.

5. Fukunaga, Keinsosuke, Introduction to Statistical Pattern Recognition, Academic Press, New York, 1972, p100-119.

6. Duda, Richard O. and Peter E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973, p98-105.

7. EG&G Reticon, R5620 Universal Active Filter Data Sheet, 1982.

APPENDIX A

TYPICAL WORDS

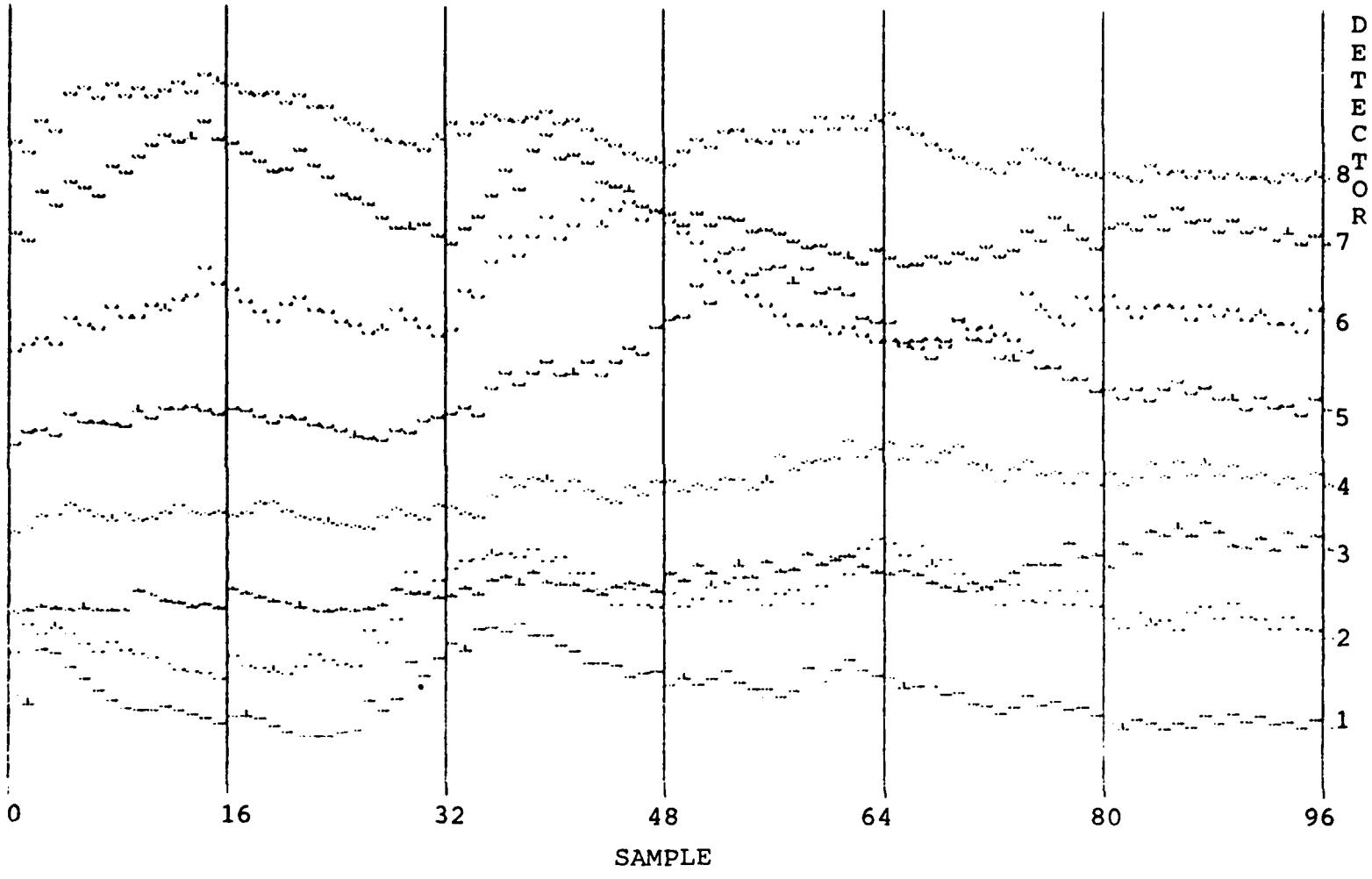


FIGURE 10. TYPICAL 'ZERO'

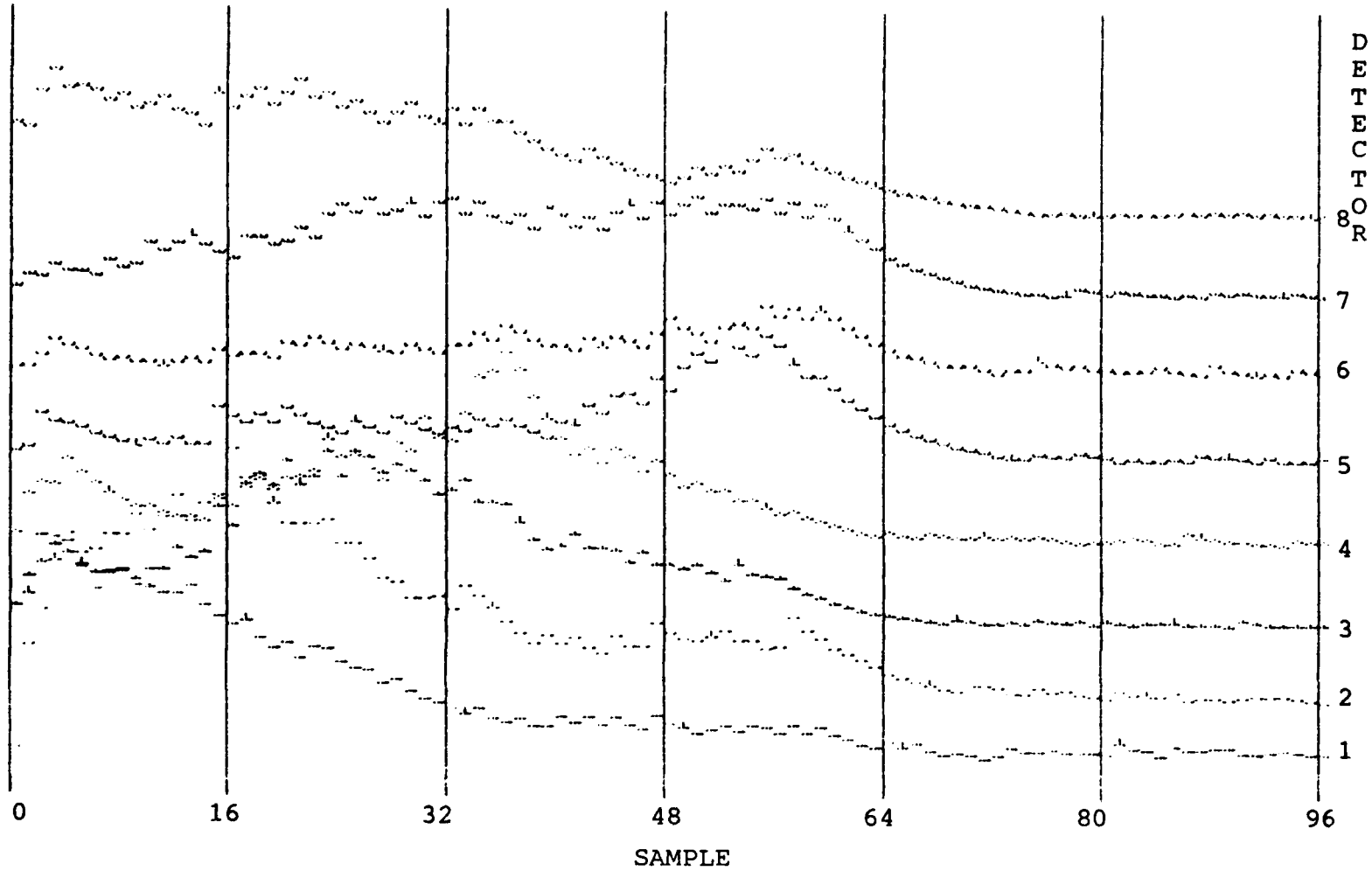


FIGURE 11. TYPICAL 'ONE'

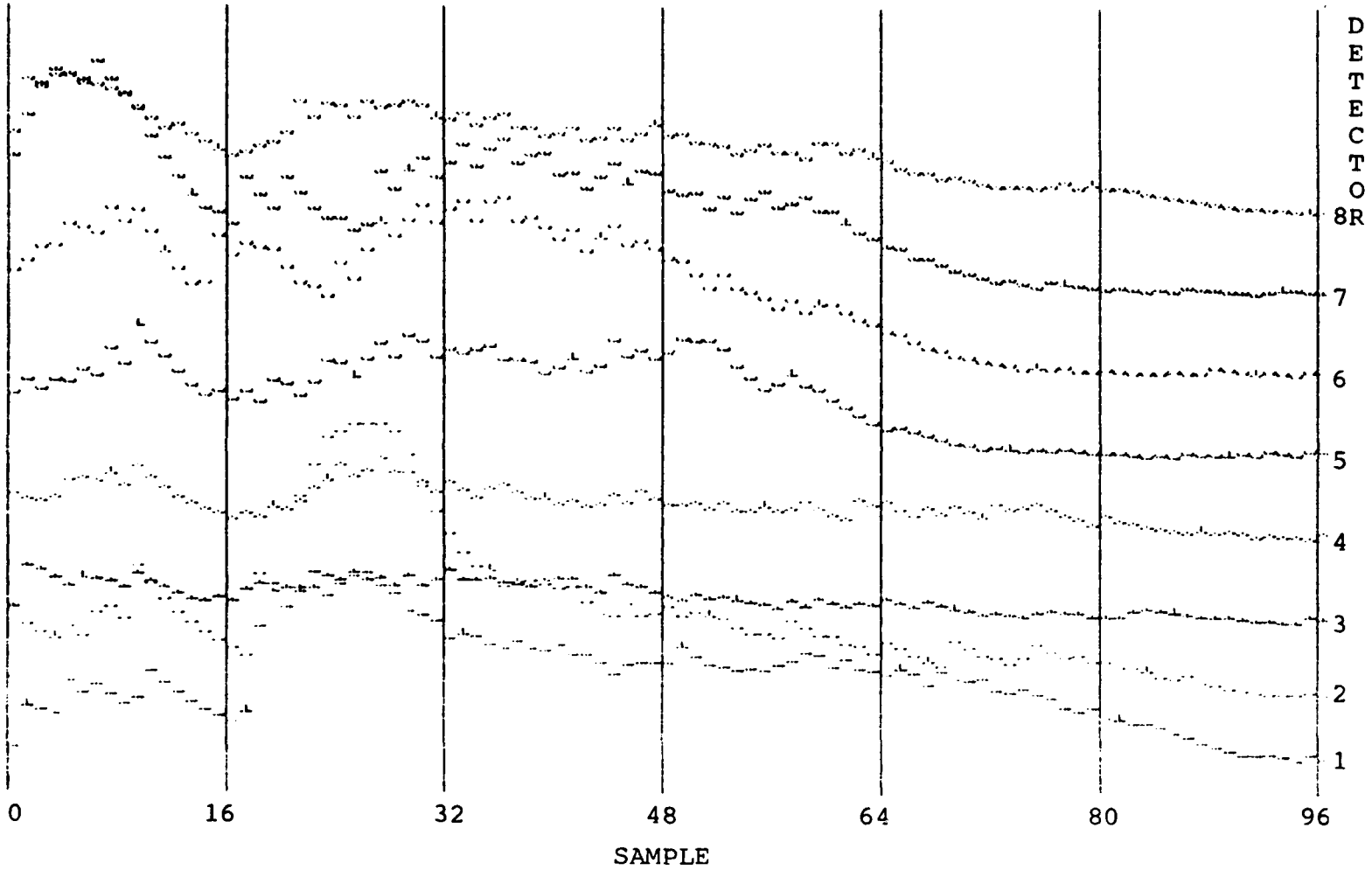


FIGURE 12. TYPICAL 'TWO'

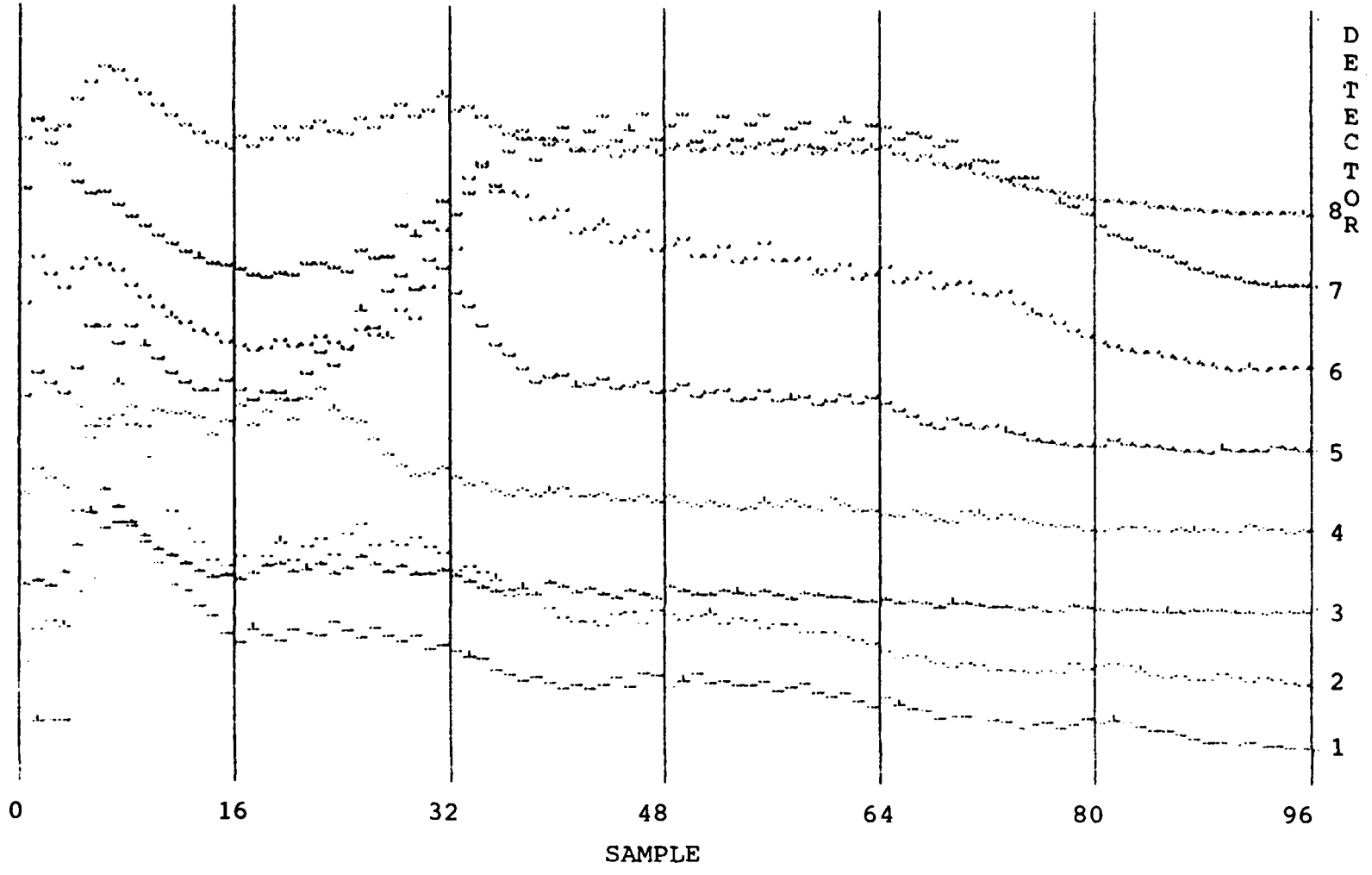


FIGURE 13. TYPICAL 'THREE'

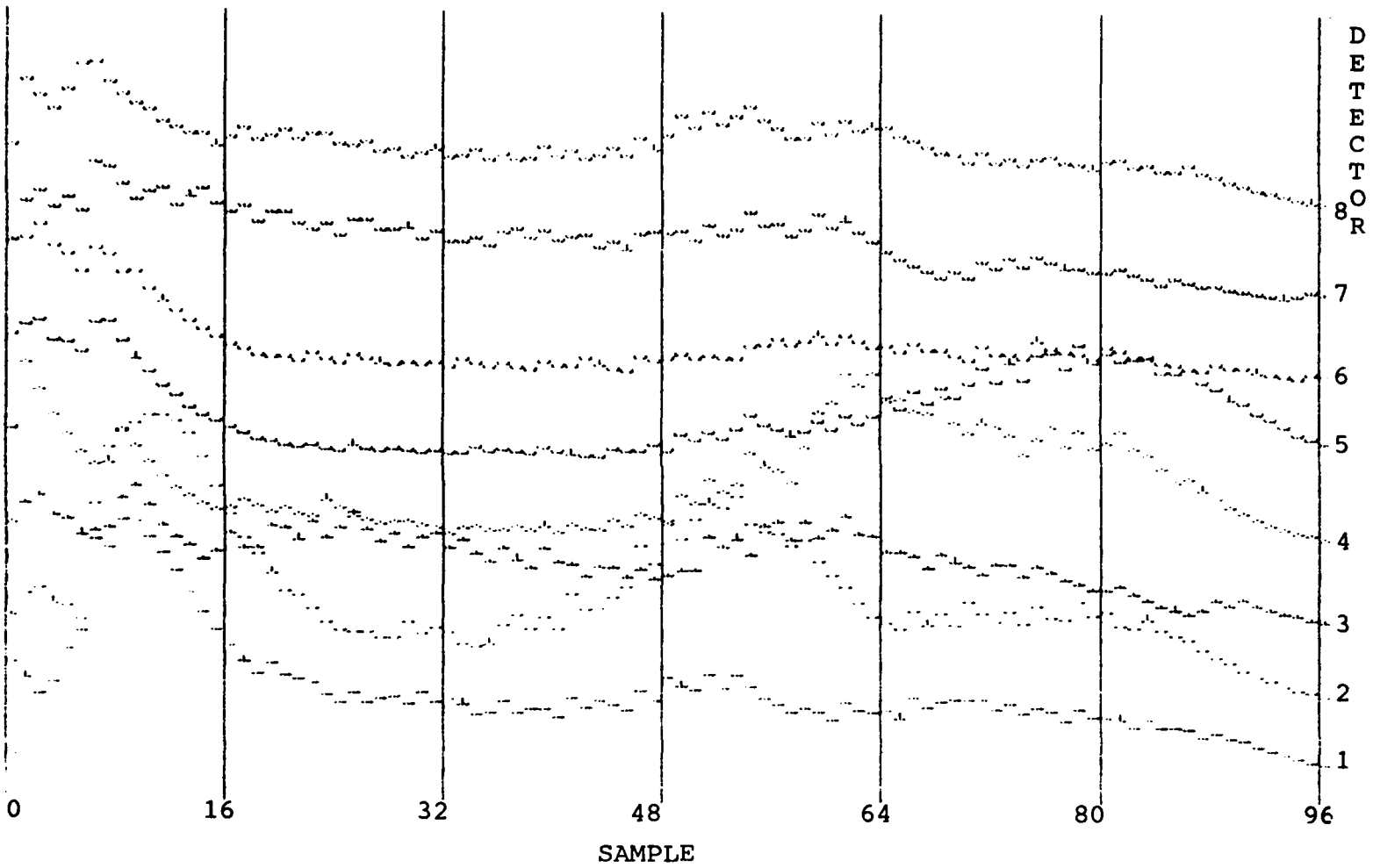


FIGURE 14. TYPICAL 'FOUR'

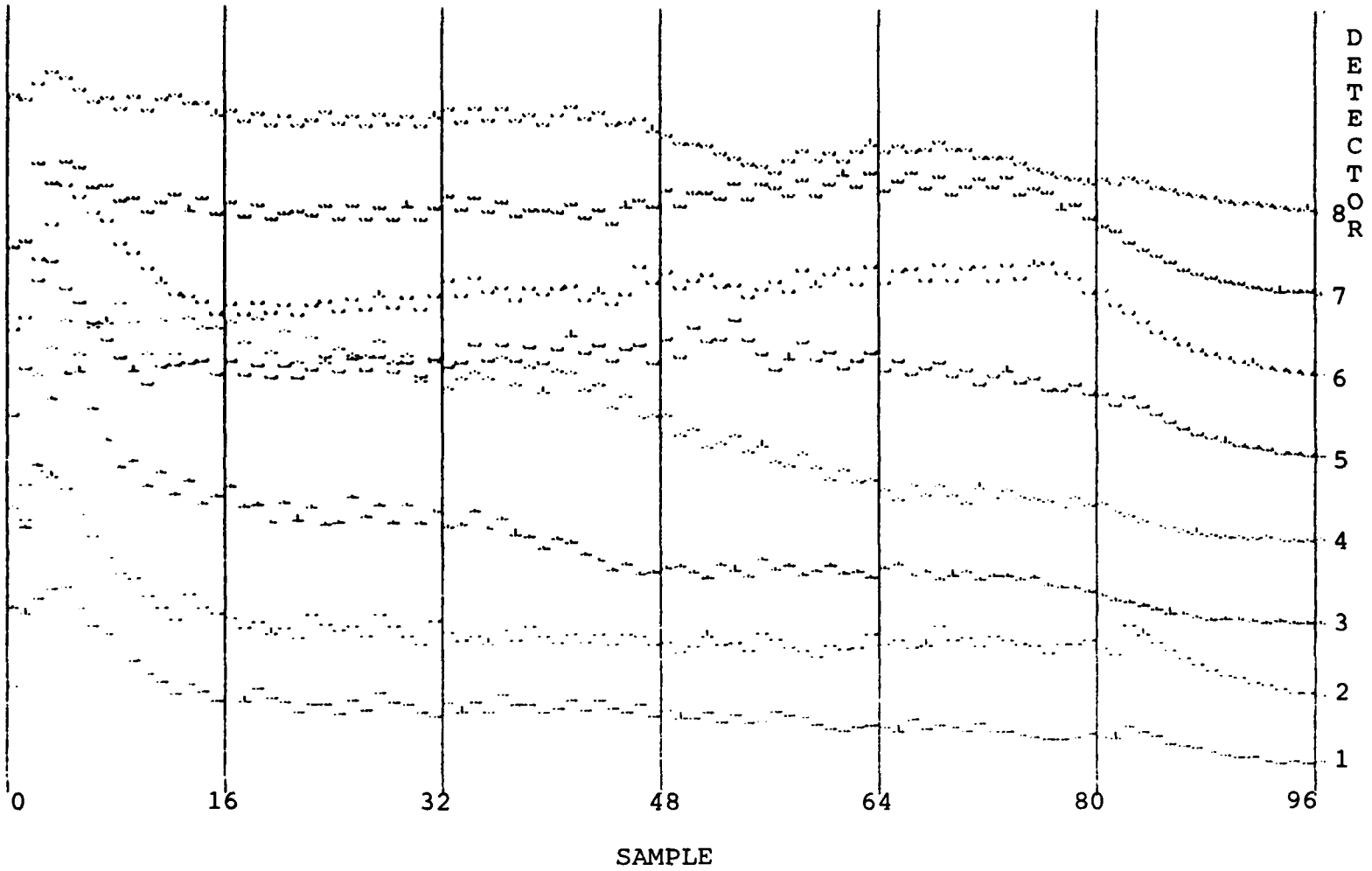


FIGURE 15. TYPICAL 'FIVE'

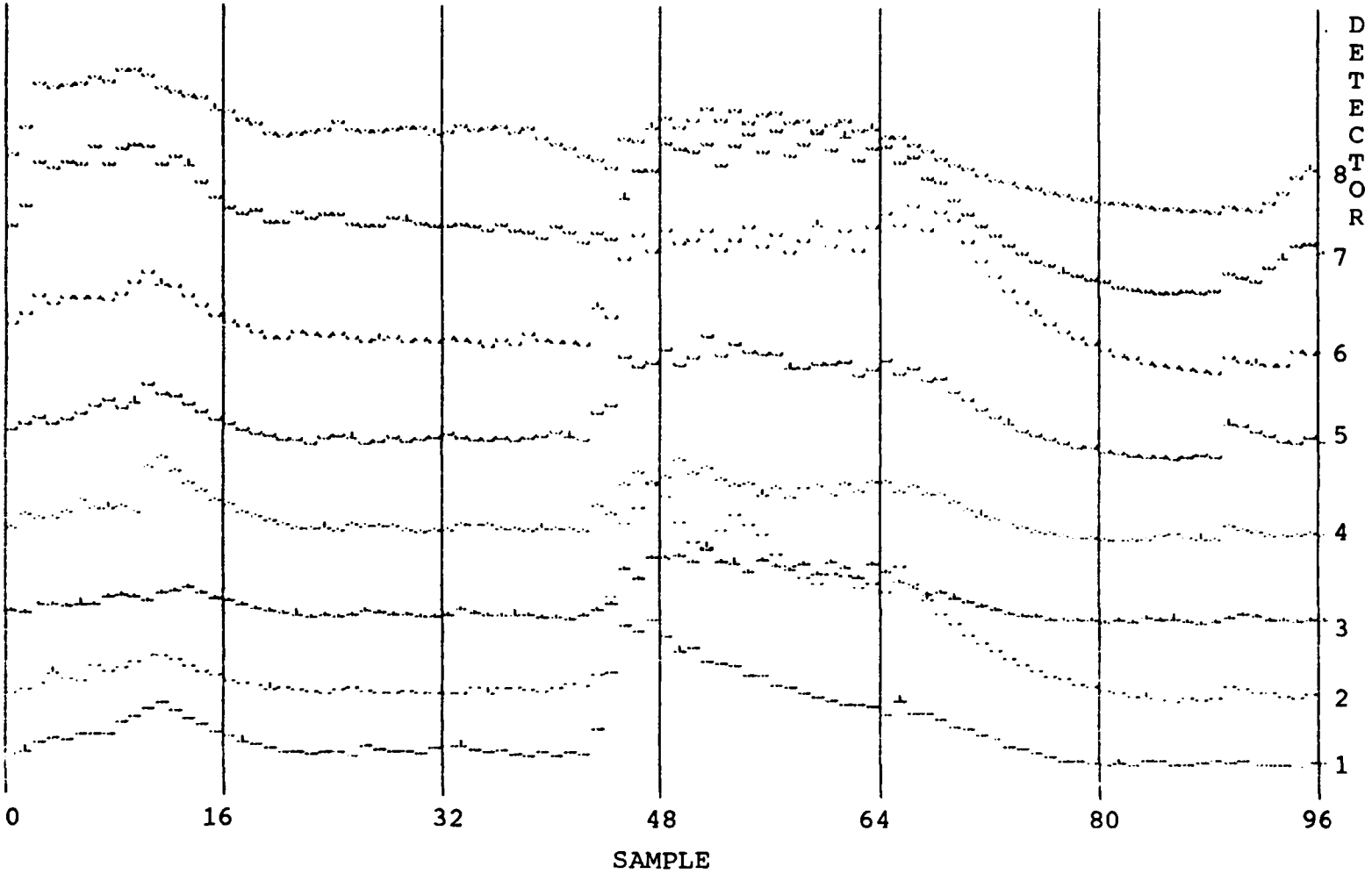


FIGURE 16. TYPICAL 'SIX'

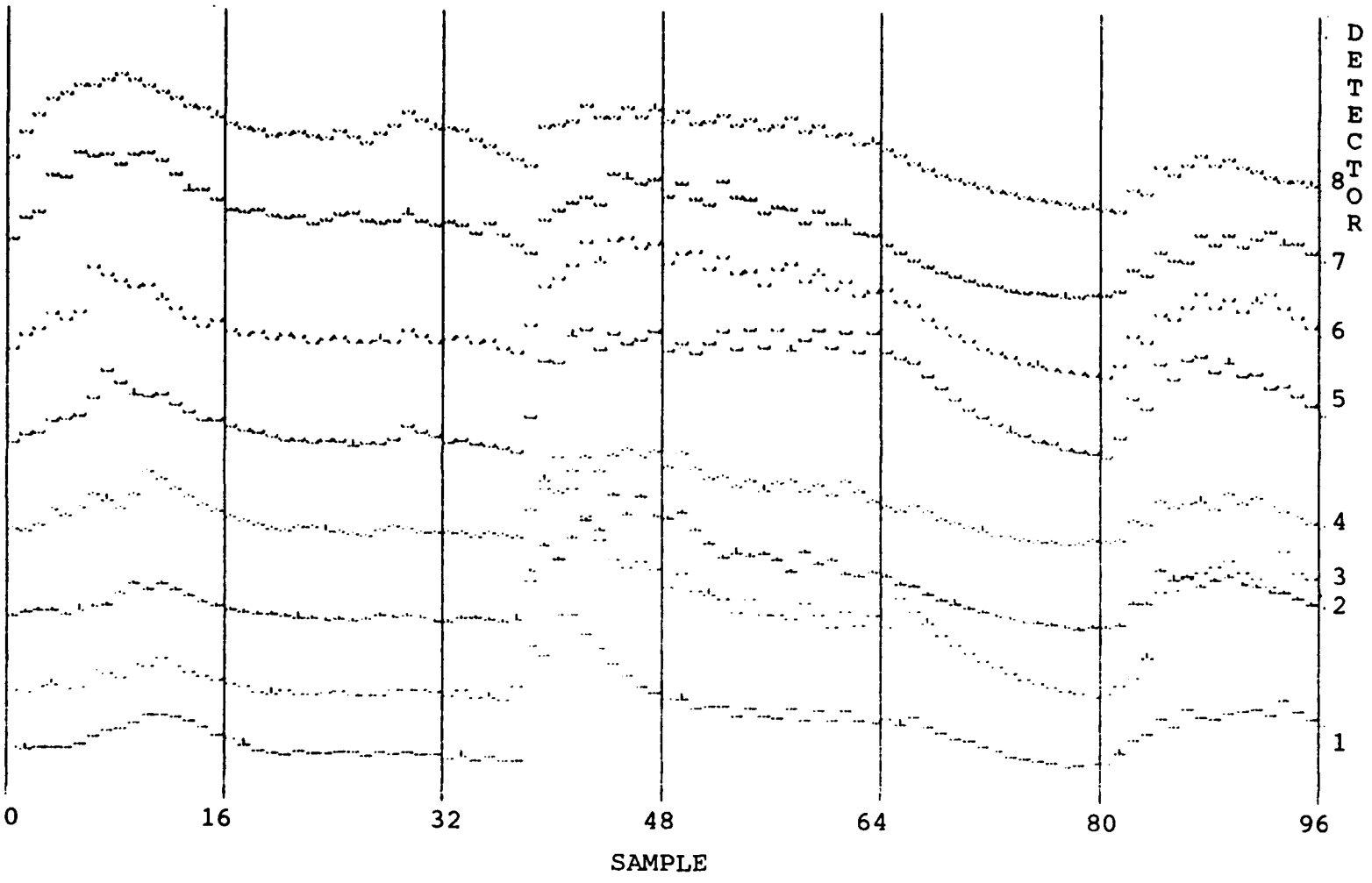


FIGURE 17. TYPICAL 'SEVEN'

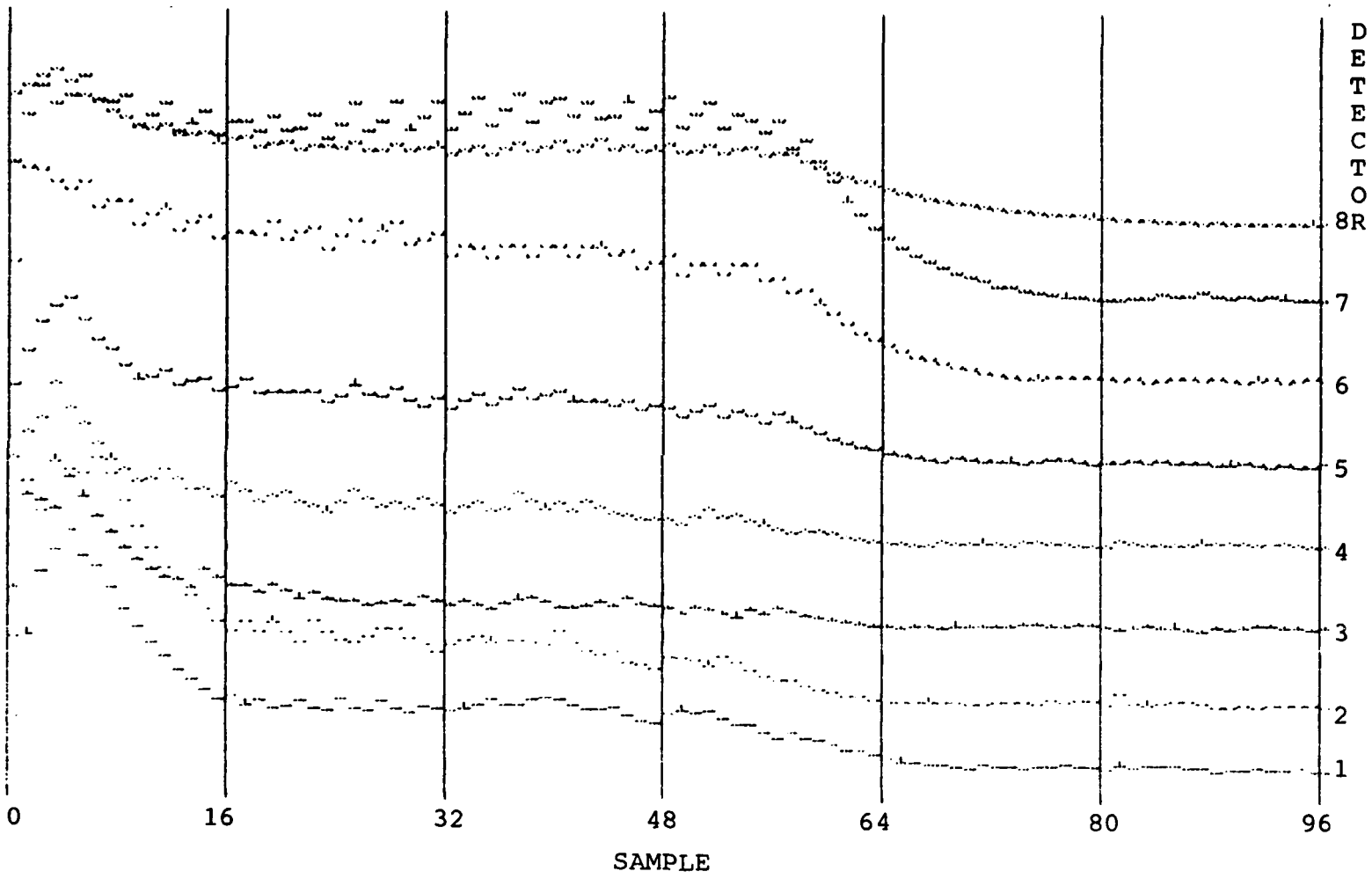


FIGURE 18. TYPICAL 'EIGHT'

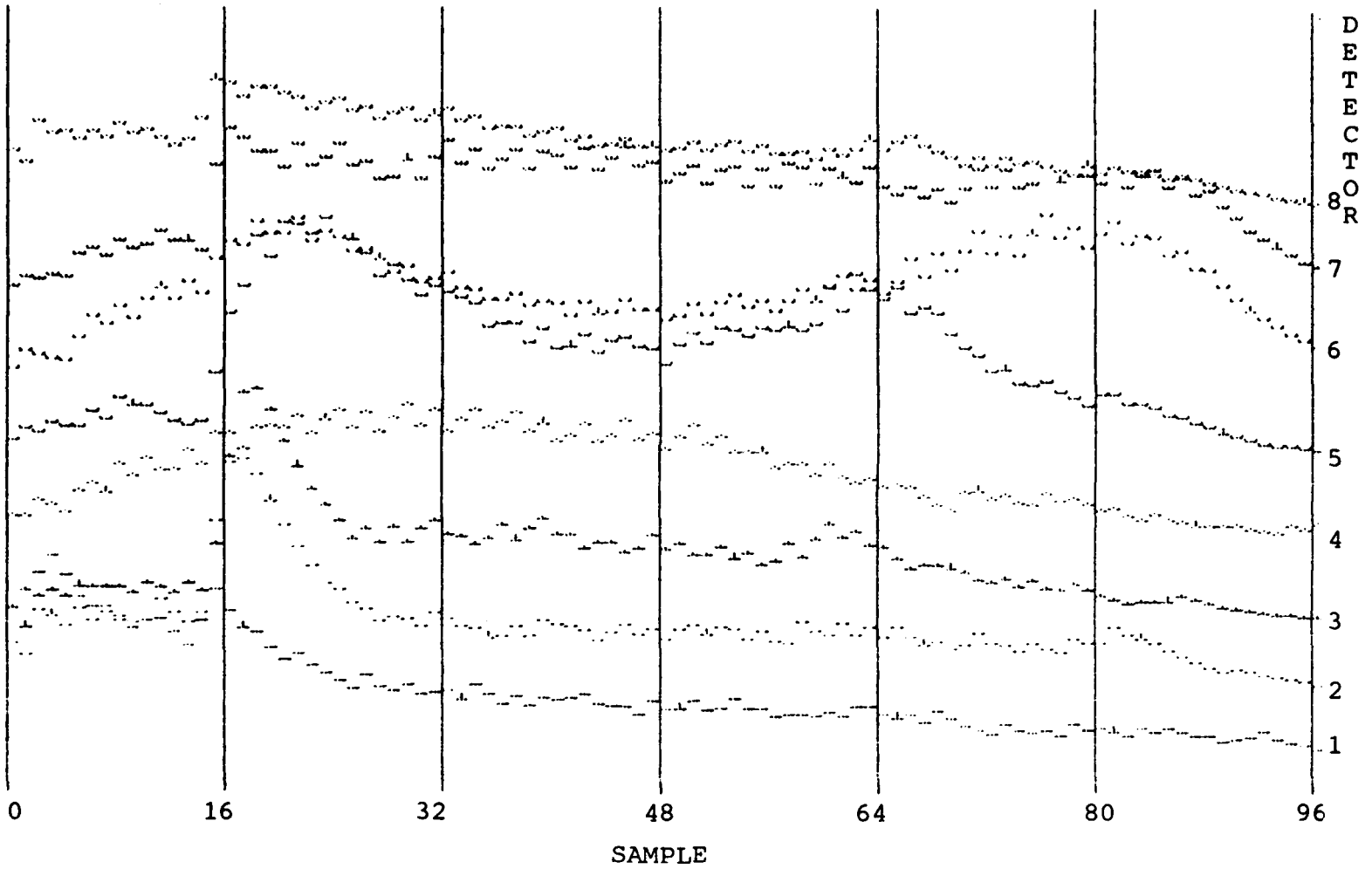


FIGURE 19. TYPICAL 'NINE'

APPENDIX B

PARTS LIST

PARTS LIST

AGC

			ohms
C1	.1uF	R1	1K
C2	.1uF	R2	100K
C3	1uF	R3	1K
C4	.1uF	R4	1K
C5	10uF	R5	10K POT
		R6	1K
D1	1N4148	R7	10K

IC1 LM348
IC2 LM370

CLOCK GENERATOR

			ohms
IC3	7404	R8	270
IC4	74LS393	R9	270

XTAL 1MHz

ENVELOPE DETECTOR

C6 .1uF
C7 1uF

D2 1N4148

IC5 LM348

	ohms
R10	1K
R11	10K
R12	1K
R13	33K

FILTER

RETICON R5620

TABLE 18. PARTS LIST

APPENDIX C

CONFUSION MATRICES

```
0 50000000000000000000000000000000000000000000000000
1 05000000000000000000000000000000000000000000000000
2 00500000000000000000000000000000000000000000000000
3 00050000000000000000000000000000000000000000000000
4 00004000000000000000000000000000000000000000100000
5 00000400000000010000000000000000000000000000000000
6 00000050000000000000000000000000000000000000000000
7 00000005000000000000000000000000000000000000000000
8 00000000500000000000000000000000000000000000000000
9 00000000050000000000000000000000000000000000000000
0 00000000040000000000000000000000000000000000000001
1 00000000004000000001000000000000000000000000000000
2 00000000000041000000000000000000000000000000000000
3 00000000000011000000000000000000000000000002000001
4 00001000000002000000000000000010010000000000000000
5 00000000010001030000000000000000000000000000000000
6 00000000000000004000000000000000001000000000000000
7 00000000000000000400000000000000000000000000000100
8 00000000000010000040000000000000000000000000000000
9 00000000000000000004000000000000000000000000000001
0 00000000000000000004000000000000000000000000001000
1 00000000000001000000040000000000000000000000000000
2 00000000000000000000050000000000000000000000000000
3 00000000000010000000004000000000000000000000000000
4 00001000000000000000003000000000000000000100000000
5 00000000000010000000000040000000000000000000000000
6 00000010000000000000000004000000000000000000000000
7 00000000000000000000000005000000000000000000000000
8 00000000000000000000000005000000000000000000000000
9 00000000010000000000000004000000000000000000000000
0 00000000000000000001000000001000002010000000000000
1 00000000000000000000000000003000000010100000000000
2 00000000000010000000000000003100000000000000000000
3 00000000000000000000000000005000000000000000000000
4 00000000000000000000000000005000000000000000000000
5 00000000000001000000000000000020002000000000000000
6 00000000000000001000000000000000040000000000000000
7 00000000000000000000000000000000401000000000000000
8 00000000000000000000000000000005000000000000000000
9 00000000000000000000000000000050000000000000000000
0 00000000000000000000000000000500000000000000000000
1 00000000000000000000000000000500000000000000000000
2 00000000000012000010000000000000000000000010000000
3 00000000000000000000000000000000000000000005000000
4 00001000000000000000000000000000000000000400000000
5 00000000000000010000000000000000000000000040000000
6 00000000000000000000000001000000000000000004000000
7 00000000000000000000000000000000000000000000000500
8 00000000000001000000000000000000000000100000000030
9 00000000000000000000000000000000000000000000000005
```

FIGURE 20. TRAINING DATA SPEAKERS 1,2,3,4,5

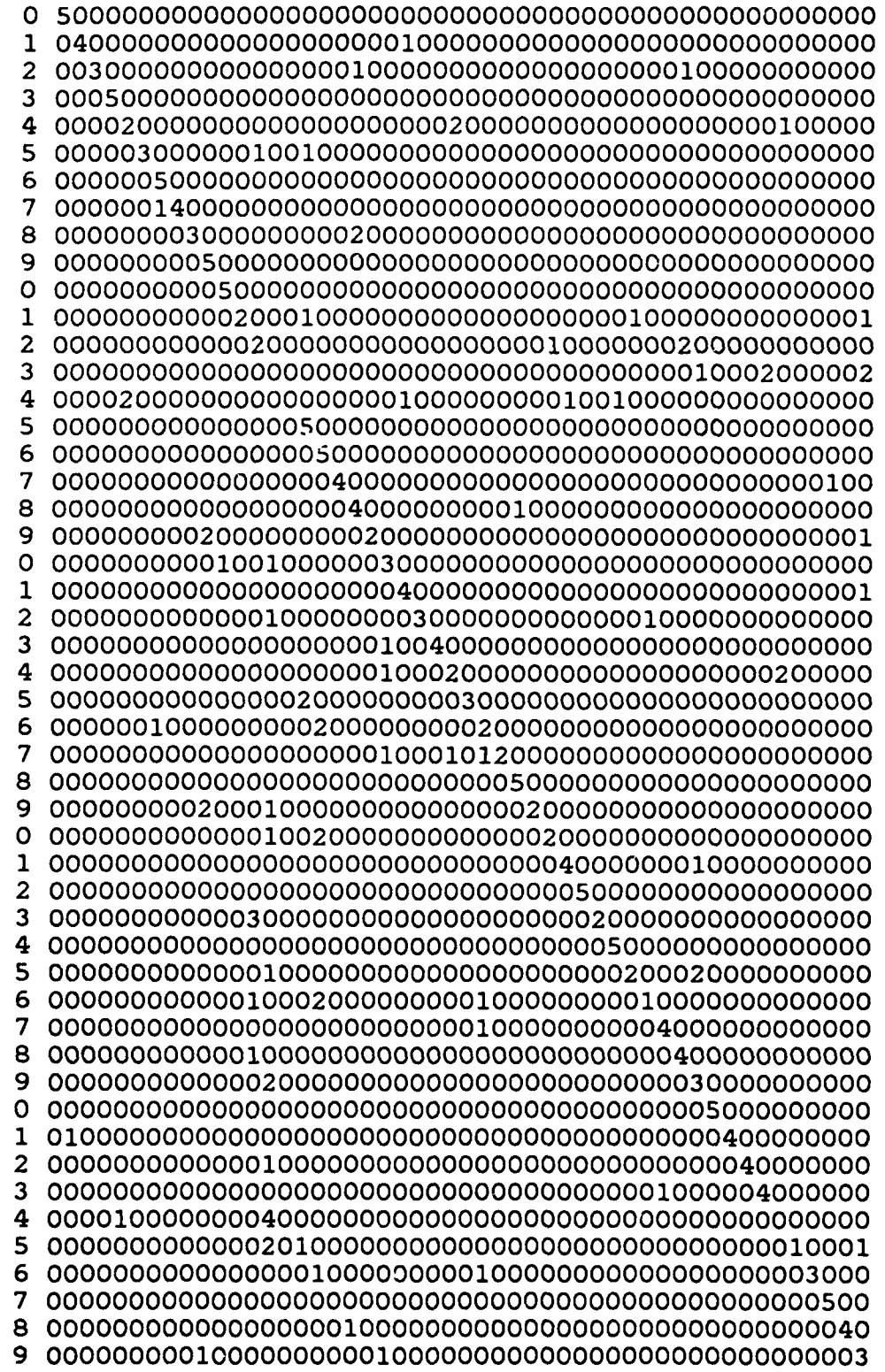


FIGURE 21. CONTROL DATA SPEAKERS 1,2,3,4,5

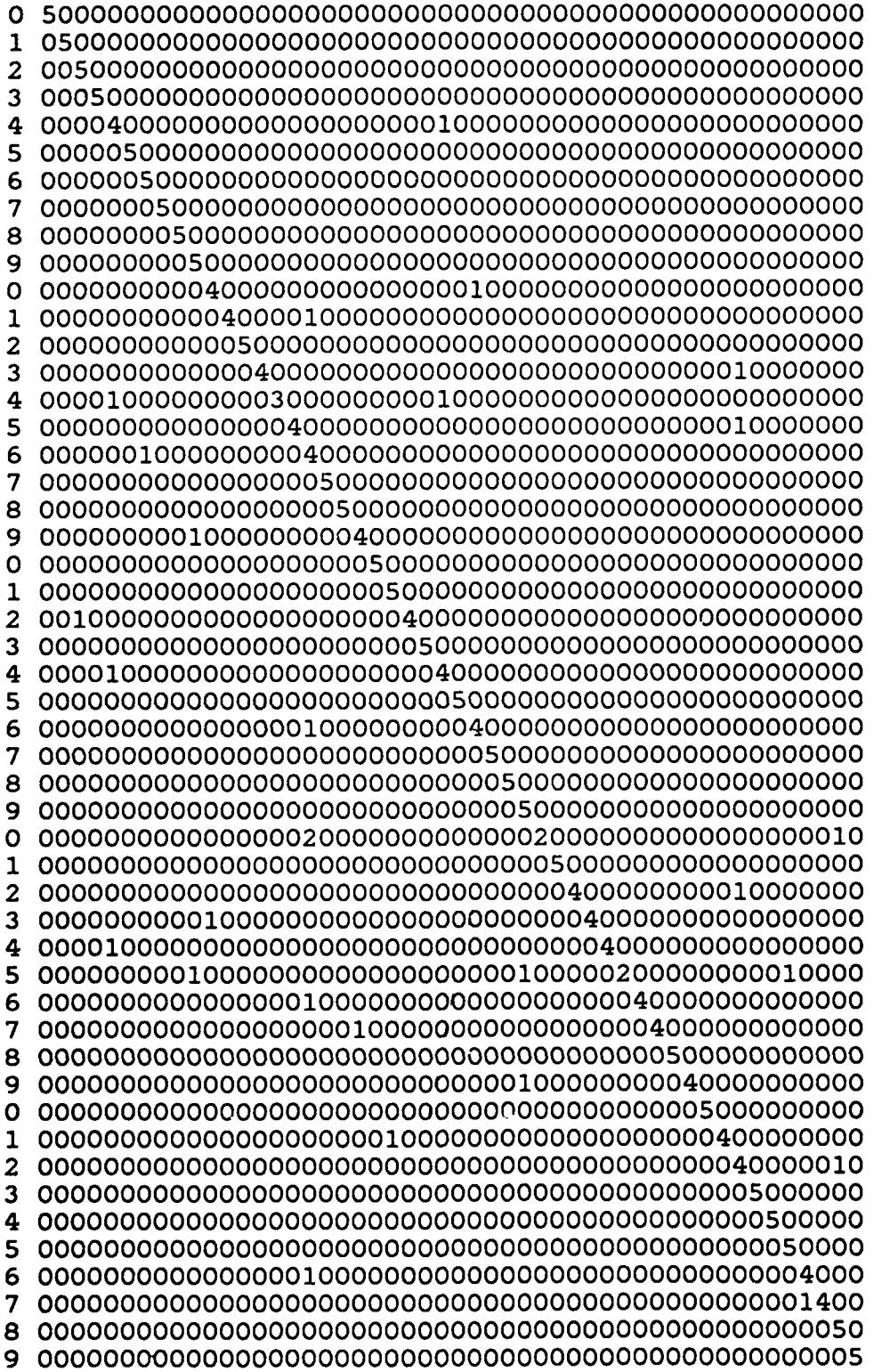


FIGURE 22. TRAINING DATA SPEAKERS 1,3,5,8,9

```
0 5000000000000000000000000000000000000000000000000000
1 0400000000010000000000000000000000000000000000000000
2 0040000000000000000000000001000000000000000000000000
3 0005000000000000000000000000000000000000000000000000
4 0000200000000010000000010000010000000000000000000000
5 0000030000000000000000000000010000000000010000000000
6 0000005000000000000000000000000000000000000000000000
7 0000001400000000000000000000000000000000000000000000
8 0000000050000000000000000000000000000000000000000000
9 0000000005000000000000000000000000000000000000000000
0 0000000003000000001000000000000000000010000000000000
1 0000000000040000000000000001000000000000000000000000
2 0000000000040001000000000000000000000000000000000000
3 0000000001004000000000000000000000000000000000000000
4 0000000001000200000000100000100000000000000000000000
5 0000010000000003000000001000000000000000000000000000
6 0000001000000000400000000000000000000000000000000000
7 0000000000000010020000000000000000010000000001000000
8 0000000000000000050000000000000000000000000000000000
9 0000000003000000002000000000000000000000000000000000
0 0000000000000000000500000000000000000000000000000000
1 0100000000000000000040000000000000000000000000000000
2 0000000000000000000005000000000000000000000000000000
3 0000000000000000000005000000000000000000000000000000
4 0000300000010000000000010000000000000000000000000000
5 0000010000000000000000002000100000000000000000000001
6 0000000000000000100000000400000000000000000000000000
7 0000000000000000000000000050000000000000000000000000
8 0000000000000000000000000005000000000000000000000000
9 0000000001000000000000000000300001000000000000000000
0 0000000000000000000000000000020000010000000000000020
1 0000000000010000000000000000003000000001000000000000
2 0000000000000000000100000000030000000010000000000000
3 0000000001000000000000000000100300000000000000000000
4 0000000000010000000100000001000020000000000000000000
5 0000010001000000000000000000010000020000000000000000
6 0000002000000000100000000000000000002000000000000000
7 0000000000000000000000000000001000000300001000000000
8 0000001000000000000000000000000000000040000000000000
9 0000000000000000000000000000011000000003000000000000
0 0000000000000000000000000000000000000004000000100000
1 0000000000000000000000001000000000000000300010000000
2 0000000000000000000000000000020000000000020001000000
3 0000000000000000000000000000000000000002003000000000
4 0000000000000000000000000000000000000000500000000000
5 0000000000000000000000001000000000000000040000000000
6 0000001000000000000000000000000000000001000003000000
7 0000000000000000000000000000000000000000011300000000
8 0000000000000000000000000000000000000000000050000000
9 0000000000000000000000000000000000000000000005000000
```

FIGURE 23. CONTROL DATA SPEAKERS 1,3,5,8,9

```
0 4000000000000010000000000000000000000000000000000000
1 0500000000000000000000000000000000000000000000000000
2 0050000000000000000000000000000000000000000000000000
3 0005000000000000000000000000000000000000000000000000
4 0000500000000000000000000000000000000000000000000000
5 0000030001000000000000000000000000000010000000000000
6 0000004000000000000000000000000000000000000000001000
7 0000000500000000000000000000000000000000000000000000
8 0000000050000000000000000000000000000000000000000000
9 0000000005000000000000000000000000000000000000000000
0 0000000004000000000000000000000000000000000000001000
1 0000000000050000000000000000000000000000000000000000
2 0000000000005000000000000000000000000000000000000000
3 0000000000004000001000000000000000000000000000000000
4 0000100000000040000000000000000000000000000000000000
5 0000000000000004000000000000000000010000000000000000
6 0000000000000000400000000000000000000000000000001000
7 0000000000000000050000000000000000000000000000000000
8 0000000000000000005000000000000000000000000000000000
9 0000000000000000000500000000000000000000000000000000
0 0000000000000000000050000000000000000000000000000000
1 0000000000000000000005000000000000000000000000000000
2 0000000000000000000000500000000000000000000000000000
3 0000000000000000000000050000000000000000000000000000
4 0000000000000000000000040000000010000000000000000000
5 0000000000000000000000005000000000000000000000000000
6 0000000000000000000000002030000000000000000000000000
7 0000000000000000000000000200000000010100000001000
8 0000000000000000000000000005000000000000000000000000
9 0000000000000000000000000000400000000010000000000
0 0000000000000000000000000000050000000000000000000000
1 0000000000000000000000000000005000000000000000000000
2 0000000000000000000000000000000410000000000000000000
3 0000000000000000000000000000000040000000000000010
4 0000000000000000000000000000000005000000000000000000
5 0000000000000000000000000000000005000000000000000000
6 0000000200000000000000000000000010010000000001000
7 0000000000000000000000000000000000000040000000001000
8 0000000000000000000000000000000000000005000000000000
9 0000000000000000000000000000001000000000400000000000
0 0000000000000000000000000000000000000000005000000000
1 0000000001000000000000000100000000000000030000000000
2 0010000000000000000000000000000000000000040000000
3 0000000000000000000000000000000000000000004000010
4 0000200000000000000000000000000000000000300000
5 0000000000000000000000000000000000000000050000
6 0000000100000000100000000000000000000000003000
7 0000000000000000000000000000000000000000000500
8 0000000000000000000000000000000000000000000050
9 0000000000000000000000000000000000000000000005
```

FIGURE 24. TRAINING DATA NON-UNIFORM ALGORITHM

```
0 2000000000000000000000000000002000000000000001000
1 0100010000000000000000000000001000000002000000000
2 0040000000000000000000000000001000000000000000000
3 0004000000000000000000000000001000000000000000000
4 0000300000000000000000000000001000000000100000
5 0000030000000000000000002000000000000000000000000
6 0000001000000000000000000000000000000000000004000
7 0000010300000000000000000010000000000000000000000
8 0000000040000000000000000001000000000000000000000
9 0000010003000000000000000000000000000000000010000
0 0000000002001000000000000000010000000000001000
1 0000000000000005000000000000000000000000000000000
2 0000000000005000000000000000000000000000000000000
3 0000000000004000001000000000000000000000000000000
4 0000000000003000000000000000002000000000000000
5 0000000000000050000000000000000000000000000000000
6 0000000000000003000000000000001000100000000000000
7 0000000000000000300000000000000000010000000100
8 0000000000000000050000000000000000000000000000000
9 0000000000001020002000000000000000000000000000000
0 0000000000000000005000000000000000000000000000000
1 0000000010000000000010000000000000003000000000
2 0000001000000000000002000000002000000000000000000
3 0000000000000000000004000000001000000000000000000
4 0000000000000000000004000000001000000000000000000
5 0000000000000010000000030000000010000000000000
6 0000000000000000000000000030000000000000000002000
7 0000000100000000000000000000000000001010000001100
8 0000000010000000000000000004000000000000000000000
9 0000000000000000000000000002000000003000000000
0 0000000001000000000000000004000000000000000000
1 0000000000000000000010001000001000000000000020000
2 0000000000000000000000000000005000000000000000000
3 0000000000000000000000000000005000000000000000000
4 0000000000000000000000000000002000000000300000
5 0000000000000001000000000000000004000000000000000
6 0000000000000000000000000000000030000000002000
7 0000002000000000000000000000000000012000000000000
8 0000000000000000000000000002000000003000000000000
9 0000000000000000000000001000000000004000000000
0 0000000000000000000000000000000000005000000000
1 0000000010000000000000000000000000004000000000
2 0000000000000000000000000000000000050000000
3 0001000000000000000000000000000001000000003000000
4 0000500000000000000000000000000000000000000000000
5 0000000000000000000000000000000000000000050000
6 0000001000000000000000000000000000000000004000
7 0000001000000000000000000000000000000000000400
8 0000000000000000000000000000000000001000000040
9 0000000000000000000000000000000000000000000005
```

FIGURE 25. CONTROL DATA NON-UNIFORM ALGORITHM

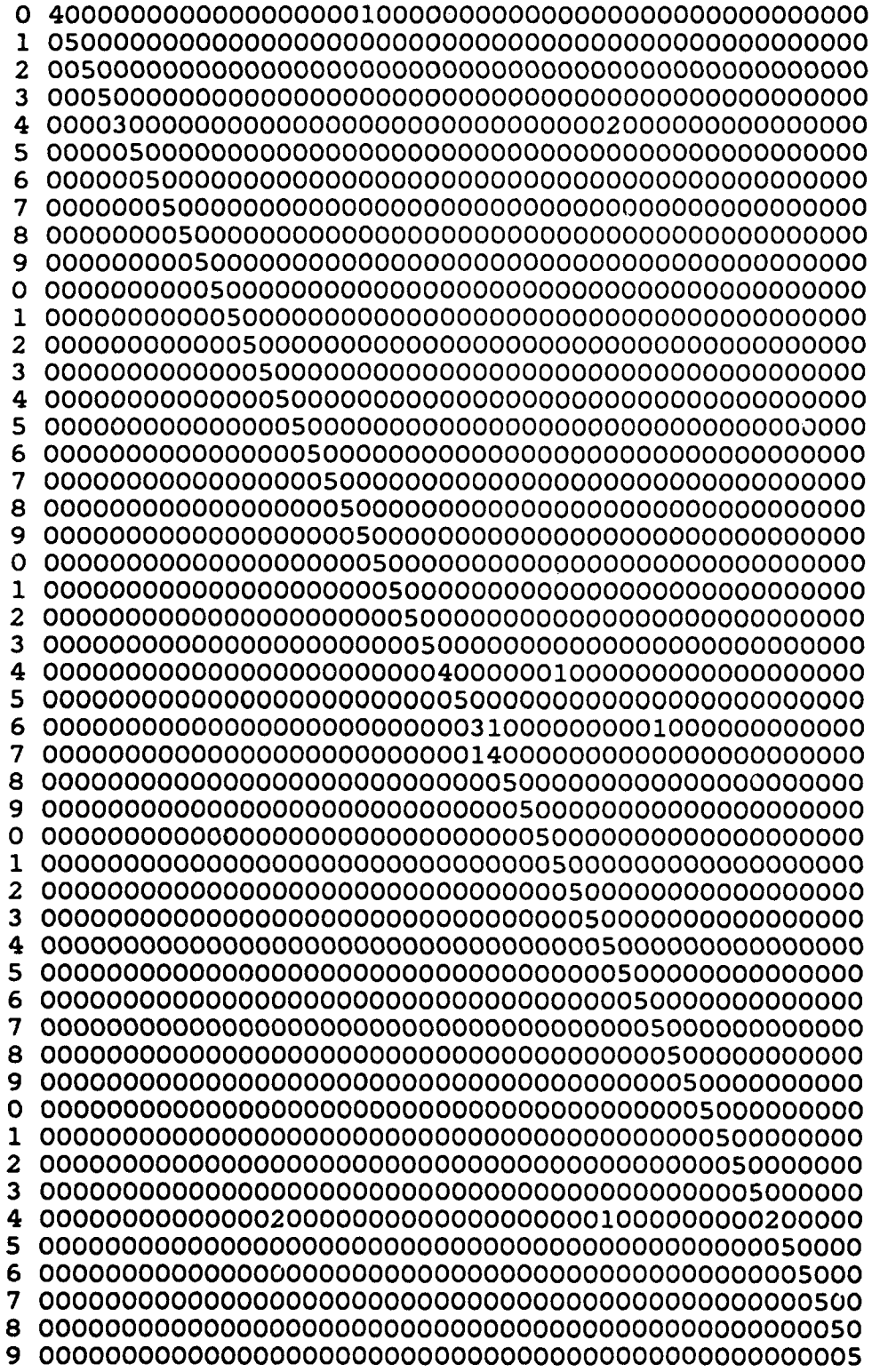


FIGURE 26. TRAINING DATA UNIFORM ALGORITHM (8 SLICE)

```
0 3000000000200000000000000000000000000000000000000
1 0500000000000000000000000000000000000000000000000
2 0050000000000000000000000000000000000000000000000
3 0004000000000000000000000000000100000000000000000
4 0000300000000000000000000000001010000000000000000
5 0000040000000000000000000000000100000000000000000
6 0000005000000000000000000000000000000000000000000
7 0000000500000000000000000000000000000000000000000
8 0000000050000000000000000000000000000000000000000
9 0000000005000000000000000000000000000000000000000
0 0000000000500000000000000000000000000000000000000
1 0000000000050000000000000000000000000000000000000
2 0000000000005000000000000000000000000000000000000
3 0000000000000500000000000000000000000000000000000
4 0000000000000040000000000000000100000000000000000
5 0000000000000040001000000000000000000000000000000
6 0000000000000000500000000000000000000000000000000
7 0000000000000000050000000000000000000000000000000
8 0000000000000000005000000000000000000000000000000
9 0000000000000000000500000000000000000000000000000
0 0000000000000000000050000000000000000000000000000
1 0000000000000000000003000000000200000000000000000
2 0000000000000000000005000000000000000000000000000
3 0000000000000000000000500000000000000000000000000
4 0000000000000000000000050000000000000000000000000
5 0000000000000000000000003000000002000000000000000
6 0000000000000000000000000130000000000000000000100
7 0000000000000000000000000013000000000000000000100
8 0000000000000000000000000005000000000000000000000
9 0000000000000000000000000000500000000000000000000
0 0000000000000000000000000000040100000000000000000
1 0000000000000000000000000000005000000000000000000
2 0000000000000000000000000000000500000000000000000
3 0000000000000000000000000000000050000000000000000
4 0000000000000030000000000000000200000000000000000
5 0000000000000000000000000000000004000000000100000
6 0000000000000000000000000000000000500000000000000
7 0000000000000000000000000000000001200000000020000
8 0000000000000000000000000000000000050000000000000
9 0000000000000000000000000000000000005000000000000
0 0000000000000000000000000000000100000000400000000
1 0000000000000000000000000000000000000000050000000
2 0000000000000000000000000000000000000000005000000
3 0001000000000000000000000000000000000000000400000
4 0000000000000030000000001000000001000000000000000
5 0000000000000000000000000000000000000000000050000
6 0000000000000000000000000000000000000000000005000
7 0000000000000000000000000000000000000000000000500
8 0000000000000000000000000000000000000000000000050
9 0000000000000000000000000000000000000000000000005
```

FIGURE 27. CONTROL DATA UNIFORM ALGORITHM (8 SLICE)

APPENDIX D

PROGRAMS

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE INIT
OBJECT MODULE PLACED IN :F1:INIT.OBJ
COMPILER INVOKED BY:  PLM80 :F1:INIT.SRC
INIT:   DO;
/* THIS PROGRAM LOADS THE SAMPLES TO BE USED AS */
/* TEMPLATES FOR THE WORD RECOGNITION PROGRAM */
CO: PROCEDURE(CHAR) EXTERNAL;
    DECLARE CHAR BYTE;
END CO;
CI: PROCEDURE BYTE EXTERNAL;
END CI;
EXIT:   PROCEDURE EXTERNAL;
END EXIT;
SAMP:   PROCEDURE EXTERNAL;
END SAMP;
DECLARE DAT0PT ADDRESS;
DECLARE DAT1PT ADDRESS;
DECLARE DAT2PT ADDRESS;
DECLARE DAT3PT ADDRESS;
DECLARE DAT4PT ADDRESS;
DECLARE DAT5PT ADDRESS;
DECLARE DAT6PT ADDRESS;
DECLARE DAT7PT ADDRESS;
DECLARE DAT0 BASED DAT0PT (128) BYTE;
DECLARE DAT1 BASED DAT1PT (128) BYTE;
DECLARE DAT2 BASED DAT2PT (128) BYTE;
DECLARE DAT3 BASED DAT3PT (128) BYTE;
DECLARE DAT4 BASED DAT4PT (128) BYTE;
DECLARE DAT5 BASED DAT5PT (128) BYTE;
DECLARE DAT6 BASED DAT6PT (128) BYTE;
DECLARE DAT7 BASED DAT7PT (128) BYTE;
DECLARE PT BYTE;
DECLARE DP BYTE;
DECLARE FAT0PT ADDRESS;
DECLARE FAT0 BASED FAT0PT (1000)BYTE;
DECLARE SAM BYTE;
DECLARE BANK BYTE;
DECLARE I BYTE;
DAT0PT=0A000H;
DAT1PT=0A080H;
DAT2PT=0A100H;
DAT3PT=0A180H;
DAT4PT=0A200H;
DAT5PT=0A280H;
DAT6PT=0A300H;
DAT7PT=0A380H;
FAT0PT=06800H;
;
```

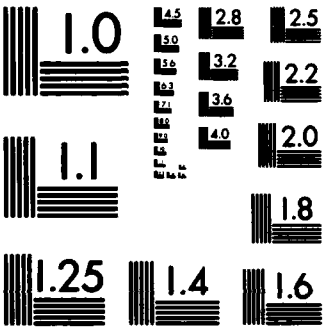
```
/* INPUT THE SAMPLE NUMBER AND THE DIGIT TO BE STORED */
START:  CALL CO(23H);
    BANK=CI AND 7FH;
    CALL CO(BANK);
    IF BANK=1AH THEN CALL EXIT;    /* CONTROL Z */
    CALL CO(3FH);
    SAM=CI AND 7FH;
    CALL CO(SAM);
    IF SAM=1AH THEN CALL EXIT;     /* CONTROL Z */
    CALL SAMP;
;
/* */
/* MOVE THE TIME SLICES TO THE CORRECT LOCATIONS */
/* IN MEMORY */
SAM=SAM-30H;
BANK=BANK-30H;
DO I=0 TO 31;
    PT=I*8;
    DP=I*4;
    FAT0((BANK*10+SAM)*256+PT)=DAT0(DP);
    FAT0((BANK*10+SAM)*256+PT+1)=DAT1(DP);
    FAT0((BANK*10+SAM)*256+PT+2)=DAT2(DP);
    FAT0((BANK*10+SAM)*256+PT+3)=DAT3(DP);
    FAT0((BANK*10+SAM)*256+PT+4)=DAT4(DP);
    FAT0((BANK*10+SAM)*256+PT+5)=DAT5(DP);
    FAT0((BANK*10+SAM)*256+PT+6)=DAT6(DP);
    FAT0((BANK*10+SAM)*256+PT+7)=DAT7(DP);
END;
GOTO START;
END INIT;
MODULE INFORMATION:
CODE AREA SIZE     = 025DH    605D
VARIABLE AREA SIZE = 0017H     23D
MAXIMUM STACK SIZE = 0006H      6D
75 LINES READ
0 PROGRAM ERROR(S)
END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE EAR
OBJECT MODULE PLACED IN :F1:EAR.OBJ
COMPILER INVOKED BY:  PLM80 :F1:EAR.SRC
EAR:    DO;
/* THIS PROGRAM IS THE WORD RECOGNITION PROGRAM */
EXIT:   PROCEDURE EXTERNAL;
END EXIT;
SAMP:   PROCEDURE EXTERNAL;
END SAMP;
DIF3:   PROCEDURE (SQPT) EXTERNAL;
    DECLARE SQPT ADDRESS;
END DIF3;
CSTS:   PROCEDURE BYTE EXTERNAL;
END CSTS;
DECLARE DAT0PT ADDRESS;
DECLARE DAT1PT ADDRESS;
DECLARE DAT2PT ADDRESS;
DECLARE DAT3PT ADDRESS;
DECLARE DAT4PT ADDRESS;
DECLARE DAT5PT ADDRESS;
DECLARE DAT6PT ADDRESS;
DECLARE DAT7PT ADDRESS;
DECLARE DAT0 BASED DAT0PT (128) BYTE;
DECLARE DAT1 BASED DAT1PT (128) BYTE;
DECLARE DAT2 BASED DAT2PT (128) BYTE;
DECLARE DAT3 BASED DAT3PT (128) BYTE;
DECLARE DAT4 BASED DAT4PT (128) BYTE;
DECLARE DAT5 BASED DAT5PT (128) BYTE;
DECLARE DAT6 BASED DAT6PT (128) BYTE;
DECLARE DAT7 BASED DAT7PT (128) BYTE;
DECLARE PT BYTE;
DECLARE DP BYTE;
DECLARE GAT0PT ADDRESS;
DECLARE GAT0 BASED GAT0PT (256) BYTE;
DECLARE I BYTE;
DECLARE LINEAR (256)ADDRESS;
GAT0PT=0C000H;
DAT0PT=0A000H;
DAT1PT=0A080H;
DAT2PT=0A100H;
DAT3PT=0A180H;
DAT4PT=0A200H;
DAT5PT=0A280H;
DAT6PT=0A300H;
DAT7PT=0A380H;
;
DO I=0 TO 255;
    LINEAR(I)=I;
END;
```

```
;
START:  CALL SAMP;
;
DO I=0 TO 31;
    PT=I*4;
    DP=I*8;
    GAT0(DP)=DAT0(PT);
    GAT0(DP+1)=DAT1(PT);
    GAT0(DP+2)=DAT2(PT);
    GAT0(DP+3)=DAT3(PT);
    GAT0(DP+4)=DAT4(PT);
    GAT0(DP+5)=DAT5(PT);
    GAT0(DP+6)=DAT6(PT);
    GAT0(DP+7)=DAT7(PT);
END;
CALL    DIF3(.LINEAR(0));
IF CSTS=0FFH THEN CALL EXIT;
GOTO START;
END EAR;
MODULE INFORMATION:
CODE AREA SIZE     = 016EH    366D
VARIABLE AREA SIZE = 0215H    533D
MAXIMUM STACK SIZE = 0002H      2D
65 LINES READ
0 PROGRAM ERROR(S)
END OF PL/M-80 COMPILATION
```

AD-A132 475 A COST CONSTRAINED SPEECH RECOGNITION SYSTEMS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH W F DAVIDSON AUG 83 AFIT/CI/NR-83-21T 2/2

UNCLASSIFIED F/G 17/2 NL



```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE DIFF
OBJECT MODULE PLACED IN :F1:DIF3.OBJ
COMPILER INVOKED BY:  PLM80 :F1:DIF3.SRC
DIFF:   DO;
/* THIS PROCEDURE PERFORMS THE DYNAMIC PROGRAM MATCH */
EXIT:   PROCEDURE EXTERNAL;
END EXIT;
CO: PROCEDURE(CHAR) EXTERNAL;
    DECLARE CHAR BYTE;
END CO;
DIF3:   PROCEDURE (SQPT) PUBLIC;
DECLARE N BYTE;
DECLARE PT BYTE;
DECLARE MIN(3) BYTE;
DECLARE NUM BYTE;
DECLARE NSAM BYTE;
DECLARE LP BYTE;
DECLARE P BYTE;
DECLARE R ADDRESS;
DECLARE CHAN BYTE;
DECLARE DIFF(3) ADDRESS;
DECLARE ERROR (32) ADDRESS;
DECLARE ERRC (32) ADDRESS;
DECLARE THRES(16) ADDRESS;
DECLARE TEMP ADDRESS;
DECLARE ANS (32) BYTE;
DECLARE SAMPPT ADDRESS;
DECLARE REFPT ADDRESS;
DECLARE SAMP BASED SAMPPT (128) BYTE;
DECLARE REF BASED REFPT (4096) BYTE;
DECLARE WEIGHT  (8) BYTE;
DECLARE RVAL ADDRESS;
DECLARE LPVAL ADDRESS;
DECLARE NUMVAL ADDRESS;
DECLARE SQPT ADDRESS;
DECLARE SQUARE BASED SQPT (256) ADDRESS;
DECLARE VAR1 BYTE;
DECLARE VAR2 BYTE;
DECLARE RMAX BYTE;
SAMPPT=0C000H;
REFPT=06800H;
/* */
THRES(0)=040;
THRES(1)=060;
THRES(2)=080;
THRES(3)=100;
THRES(4)=120;
THRES(5)=140;
THRES(6)=160;
```

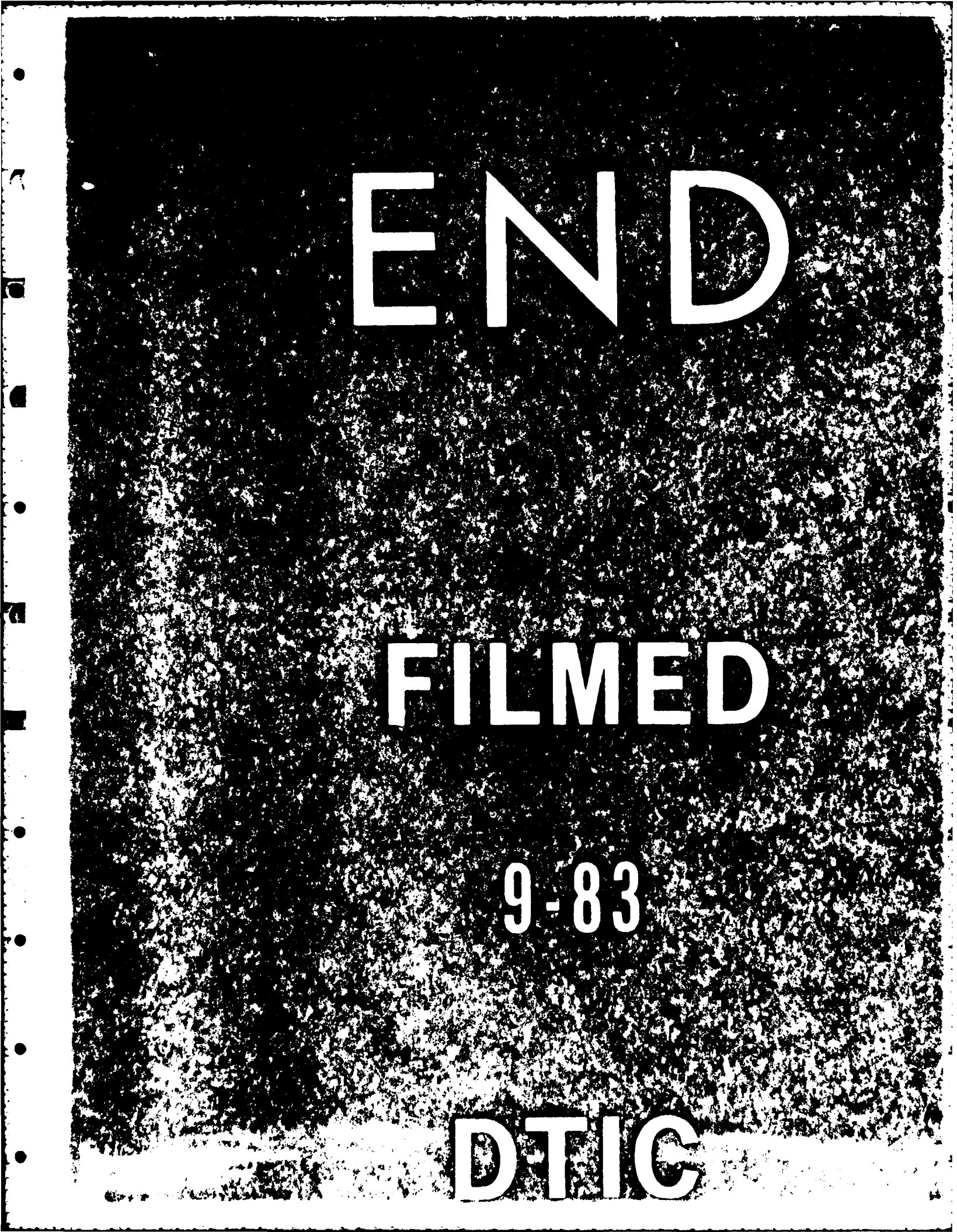
```
THRES(7)=180;
THRES(8)=200;
THRES(9)=220;
THRES(10)=240;
THRES(11)=260;
THRES(12)=280;
THRES(13)=300;
THRES(14)=320;
THRES(15)=340;
/* */
WEIGHT(0)=2;
WEIGHT(1)=2;
WEIGHT(2)=0;
WEIGHT(3)=0;
WEIGHT(4)=1;
WEIGHT(5)=1;
WEIGHT(6)=1;
WEIGHT(7)=0;
/* */
/* */
DO R=0 TO 29;
ERROR(R)=0;
ERRC(R)=0;
RVAL=SHL(R,8);
RMAX=15;
LP=0;
DO NUM=0 TO 15 BY 2;
    NUMVAL=SHL(NUM,3);
    DO P=0 TO 2;
        LPVAL=RVAL+SHL((LP+P),3);
        DIFF(P)=0;
        DO CHAN=0 TO 7;
            VAR1= REF(LPVAL+CHAN);
            VAR2=SAMP(NUMVAL+CHAN);
            IF VAR1<VAR2 THEN TEMP=VAR2-VAR1;
            ELSE TEMP=VAR1-VAR2;
            DIFF(P)=DIFF(P)+SHR(SQUARE(TEMP),WEIGHT(CHAN));
        END;
    FINI:   ;
    END;
    IF DIFF(1)<=DIFF(0) AND DIFF(1)<=DIFF(2) THEN DO;
        ERROR(R)=ERROR(R)+DIFF(1);
        LP=LP+2;
        GOTO ST;
    END;
    IF DIFF(2)<=DIFF(0) AND DIFF(2)<=DIFF(1) THEN DO;
        ERROR(R)=ERROR(R)+DIFF(2);
        LP=LP+4;
        GOTO ST;
    END;
```

```
     ELSE DO;
          ERROR(R)=ERROR(R)+DIFF(0);
          LP=LP+0;
          GOTO ST;
     END;
/* CHECK TO SEE IF YOU HAVE REACHED THE END OF THE REFERENCE */
     ST: IF NUM>=RMAX THEN GOTO TERMIN;
/* */
     IF ERROR(R)>=THRES(NUM) THEN DO;
          CALL CO(28H);
          CALL CO(30H+R);
          CALL CO(3AH);
          CALL CO(NUM+30H);
          CALL CO(29H);
          ERROR(R)=65530;
          GOTO STP;
     END;
END;
GOTO STP;
TERMIN: ERROR(R)=ERROR(R)/NUM*NSAM;
STP:    ;
/* */
END;
;
DO N=0 TO 2;
     MIN(N)=0;
     DO PT=1 TO 29;
          IF ERRC(PT)<ERRC(MIN(N)) THEN DO;
               MIN(N)=PT;
               GOTO LP;
          END;
          IF ERRC(PT)=ERRC(MIN(N)) THEN DO;
               IF ERROR(PT)<ERROR(MIN(N)) THEN DO;
                    MIN(N)=PT;
               END;
          END;
     LP: ;
     END;
     ERRC(MIN(N))=ERRC(MIN(N))+10H;
END;
;
/* */
```

```
IF ERROR(MIN(0))>65000 THEN DO;
    IF ERROR(MIN(1))>65000 THEN DO;
        IF ERROR(MIN(2))>65000 THEN DO;
            CALL CO(4EH);
            CALL CO(4FH);
            CALL CO(4EH);
            CALL CO(45H);
            GOTO F0;
        END;
    END;
END;
/* */
DO N=0 TO 1;
    IF MIN(0)>9 THEN MIN(0)=MIN(0)-10;
    IF MIN(1)>9 THEN MIN(1)=MIN(1)-10;
    IF MIN(2)>9 THEN MIN(2)=MIN(2)-10;
END;
;
IF MIN(1)=MIN(2) THEN GOTO P1;
;
CALL CO(30H+MIN(0));
GOTO F0;
P1: CALL CO(30H+MIN(1));
F0: ;
CALL CO(0DH);
CALL CO(0AH);
;
END DIF3;
END DIFF;
MODULE INFORMATION:
CODE AREA SIZE     = 047EH   1150D
VARIABLE AREA SIZE = 00EBH    235D
MAXIMUM STACK SIZE = 0006H      6D
167 LINES READ
0 PROGRAM ERROR(S)
END OF PL/M-80 COMPILATION
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER
  LINE         SOURCE STATEMENT
     1          NAME SAMP
     2          ORG      9000H
     3          PUBLIC   SAMP
     4 ;
     5 ; THIS ROUTINE INPUTS 128 SAMPLES OF EACH OF
     6 ; THE 8 CHANNELS AT 10 MS INTERVALS AFTER THE
     7 ; THRESHOLD IS REACHED
     8 ;
     9 SAMP:
    10 ST:      IN       0CFH
    11          CPI      20H       ;COMPARE CHANNEL 8 TO THRES
    12          JC       ST        ;WAIT FOR THRESHOLD
    13          MVI      D,0
    14          MVI      E,80H
    15 LP:      MVI      H,0A0H
    16          MOV      L,D
    17          IN       0C8H      ;INPUT CHANNEL 1
    18          MOV      M,A
    19          MOV      L,E
    20          IN       0C9H      ;INPUT CHANNEL 2
    21          MOV      M,A
    22          MVI      H,0A1H
    23          MOV      L,D
    24          IN       0CAH      ;INPUT CHANNEL 3
    25          MOV      M,A
    26          MOV      L,E
    27          IN       0CBH      ;INPUT CHANNEL 4
    28          MOV      M,A
    29          MVI      H,0A2H
    30          MOV      L,D
    31          IN       0CCH      ;INPUT CHANNEL 5
    32          MOV      M,A
    33          MOV      L,E
    34          IN       0CDH      ;INPUT CHANNEL 6
    35          MOV      M,A
    36          MVI      H,0A3H
    37          MOV      L,D
    38          IN       0CEH      ;INPUT CHANNEL 7
    39          MOV      M,A
    40          MOV      L,E
    41          IN       0CFH      ;INPUT CHANNEL 8
    42          MOV      M,A
    43          MVI      B,0FFH    ;DELAY
```

```
44 D1:      DCR     B
45          JNZ     D1
46          MVI     B,OFFH  ;DELAY
47 D2:      DCR     B
48          JNZ     D2
49          MVI     B,OFFH  ;DELAY
50 D3:      DCR     B
51          JNZ     D3
52          INR     D       ;INCREMENT POINTER
53          INR     E       ;INCREMENT POINTER
54          JNZ     LP      ;WAIT FOR 128 SAMPLES
55          RET
56          END
PUBLIC SYMBOLS
SAMP   A 9000
EXTERNAL SYMBOLS
USER SYMBOLS
D1     A 9035    D2     A 903B    D3     A 9041
LP     A 900B    SAMP   A 9000    ST     A 9000
ASSEMBLY COMPLETE,   NO ERRORS
```


END
FILMED
9-83
DTIC